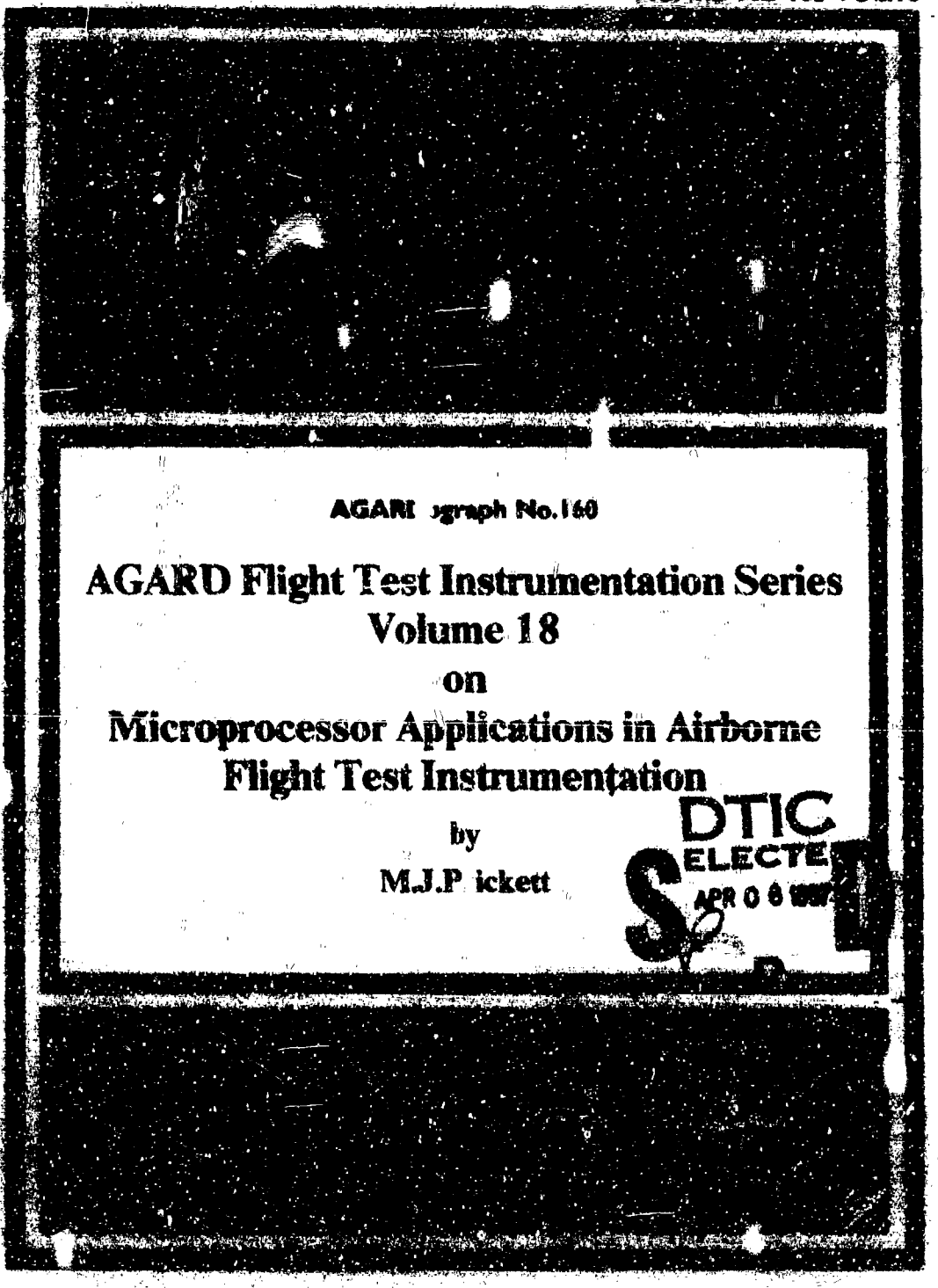
AGARD-AG-160-VOL.18



AGARDograph No.160

AGARD Flight Test Instrumentation Series Volume 18 on Microprocessor Applications in Airborne Flight Test Instrumentation

by

M.J.P ickett

AGARD-AG-160-Vol.18

NORTH ATLANTIC TREATY ORGANIZATION

ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT

(ORGANISATION DU TRAITE DE L'ATLANTIQUE NORD)

AGARDograph No.160 Vol.18

MICROPROCESSOR APPLICATIONS IN AIRBORNE FLIGHT TEST INSTRUMENTATION

by

M.J.Prickett

A Volume of the

AGARD FLIGHT TEST INSTRUMENTATION SERIES

Edited by

R.K.Bogue

This AGARDograph has been sponsored by the Flight Mechanics Panel of AGARD.

THE MISSION OF AGARD

The mission of AGARD is to bring together the leading personalities of the NATO nations in the fields of science and technology relating to aerospace for the following purposes:

- Exchanging of scientific and technical information;
- Continuously stimulating advances in the aerospace sciences relevant to strengthening the common defence posture;
- Improving the co-operation among member nations in aerospace research and development;
- Providing scientific and technical advice and assistance to the Military Committee in the field of aerospace research and development (with particular regard to its military application);
- Rendering scientific and technical assistance, as requested, to other NATO bodies and to member nations in connection with research and development problems in the aerospace field;
- Providing assistance to member nations for the purpose of increasing their scientific and technical potential;
- Recommending effective ways for the member nations to use their research and development capabilities for the common benefit of the NATO community.

The highest authority within AGARD is the National Delegates Board consisting of officially appointed senior representatives from each member nation. The mission of AGARD is carried out through the Panels which are composed of experts appointed by the National Delegates, the Consultant and Exchange Programme and the Aerospace Applications Studies Programme. The results of AGARD work are reported to the member nations and the NATO Authorities through the AGARD series of publications of which this is one.

Participation in AGARD activities is by invitation only and is normally limited to citizens of the NATO nations.

Published February 1987

ISBN 92-835-1542-0

Printed by Specialised Printing Services Limited
40 Chigwell Lane, Loughton, Essex IG10 3TZ

PREFACE

Since its founding in 1952, the Advisory Group for Aerospace Research and Development has published, through the Flight Mechanics Panel, a number of standard texts in the field of flight testing. The original Flight Test Manual was published in the years 1954 to 1956. The Manual was divided into four volumes: I. Performance, II. Stability and Control, III. Instrumentation Catalog, and IV. Instrumentation Systems.

As a result of developments in the field of flight test instrumentation, the Flight Test Instrumentation Group of the Flight Mechanics Panel was established in 1968 to update Volumes III and IV of the Flight Test Manual by the publication of the Flight Test Instrumentation Series, AGARDograph 160. In its published volumes AGARDograph 160 has covered recent developments in flight test instrumentation.

In 1978, the Flight Mechanics panel decided that further specialist monographs should be published covering aspects of Volume I and II of the original Flight Test Manual, including the flight testing of aircraft systems. In March 1981, the Flight Test Techniques Group was established to carry out this task. The monographs of this Series (with the exception of AG 237 which was separately numbered) are being published as individually numbered volumes of AGARDograph 300. At the end of each volume of AGARDograph 300 two general Annexes are printed; Annex 1 provides a list of the volumes published in the Flight Test Instrumentation Series and in the Flight Test Techniques Series. Annex 2 contains a list of handbooks that are available on a variety of flight test subjects, not necessarily related to the contents of the volume concerned.

Special thanks and appreciation are extended to Mr F.N.Stoliker (US), who chaired the Group for two years from its inception in 1981, established the ground rules for the operation of the Group and marked the outlines for future publications.

In the preparation of the present volume the members of the Flight Test Techniques Group listed below have taken an active part. AGARD has been most fortunate in finding these competent people willing to contribute their knowledge and time in the preparation of this volume.

Bogue, R.K.	(editor)	NASA/US.
Borek, R.W.		NASA/US.
Bothe, H.		DFVLR/GE.
Bull, E.J.		A & AEE/UK.
Carabelli, R.		SAI/IT.
Galan, R.C.		CEV/FR.
Lapchine, N.		CEV/FR.
Norris, E.J.		A & AEE/UK.
Phillips, A.D.		AFFTC/US.
Pool, A.		NLR/NE.
Van Doorn, J.T.M.		NLR/NE.

C.E.ADOLPH AFFTC/US
Member, Flight Mechanics Panel
Chairman, Flight Test
Techniques Group.

CONTENTS

		Page
PREFACE		iii
SUMMARY		1
1.	INTRODUCTION	1
1.1	BRIEF HISTORY	1
1.2	FUTURE POSSIBILITIES	2
2.	GENERAL MICROPROCESSOR APPLICATIONS IN FLIGHT TEST INSTRUMENTATION	2
2.1	BRIEF APPLICATIONS	2
2.2	SENSOR AND BUS CONDITIONING	3
2.3	FLIGHT-TEST SYSTEM AND DISPLAY CONTROL	4
2.4	SENSOR CALIBRATION	4
2.5	MICROPROCESSOR BENEFITS TO FLIGHT-TEST SYSTEMS	5
3.	MICROPROCESSOR-BASED SYSTEM-DESIGN CONSIDERATIONS	5
3.1	MICROPROCESSOR SYSTEM ARCHITECTURE	6
3.1.1	Memory	7
3.1.2	Special Purpose Registers	8
3.1.3	Program Execution Control	9
3.1.4	The Stack	9
3.1.5	Instruction Cycle	9
3.1.6	Interrupts	9
3.1.7	The Microprocessor Chip Set	10
3.1.8	Direct Memory Access	11
3.1.9	Bus Buffering/Demultiplexing	12
3.1.10	Three-State Devices	12
3.1.11	Memory-Mapped I/O and Isolated I/O	13
3.1.12	Microprocessor Instructions	13
3.1.13	Multiprocessing or Networking	14
3.2	MICROPROCESSOR SELECTION CONSIDERATIONS	14
3.2.1	Application Considerations	14
3.2.2	Implementation Considerations	15
3.2.3	Technology Choices	15
3.2.4	General Selection Considerations	15
3.3	SOFTWARE DEVELOPMENT	16
3.3.1	Software Development Stages	16
3.3.2	Hardware and Software Tradeoffs	17
3.4	MICROPROCESSOR SYSTEM DESIGN-SUPPORT EQUIPMENT	18
3.4.1	The Logic Analyzer	19
3.4.2	Emulators	19
3.4.3	Microprocessor Development Systems	20
3.5	SYSTEM ENVIRONMENT AND RELIABILITY	20
3.5.1	Environment Estimation	20
3.5.2	Reliability	21
3.5.3	Failure Mechanisms	21
3.5.4	Temperature Effects	23
4.	SELECTED APPLICATIONS OF AIRBORNE MICROPROCESSORS IN FLIGHT-TEST PROGRAMS	23
4.1	AIRBORNE DATA SYSTEM WITH ONBOARD REAL-TIME ANALYSIS (LOCKHEED GEORGIA)	23
4.1.1	Microprocessor-Related Functions	25
4.2	AIRFRAME FLUTTER TESTING (MCDONNELL AIRCRAFT)	25
4.2.1	Flutter Excitation (Flight-Testing Background)	25
4.2.2	Characteristics of the Flutter Exciter	27
4.2.3	The Microprocessor	27
4.2.4	Firmware	28
4.3	IN-FLIGHT FAILURE SIMULATION SYSTEM (AAEE, UK)	28
4.3.1	Flight Safety Precautions	30
4.3.2	The Microprocessor	30
4.4	ARINC DISPLAY SYSTEM (NLR, NETHERLANDS)	31
4.5	STALL-SPEED WARNING SYSTEM (NASA)	32
4.5.1	Speed Calculations	32
4.5.2	General Hardware Description	33
5.3	Single-Board Microcomputer	34

		Page
4.5.4	8231A Arithmetic Processor (Coprocessor)	36
4.5.5	EEPROM	36
4.5.6	Voice Synthesizer	36
4.5.7	System Software	36
4.6	ARINC-429 TELEMETRY INTERFACE (DFVLR, WEST GERMANY)	39
4.6.1	The Microprocessor	39
4.6.2	Additional Airborne Microprocessor-Based Systems	40
5.	COMMERCIAL MICROPROCESSORS AND SUPPORT DEVICES	40
6.	REFERENCES	41
APPENDIX A: A Limited Glossary of Microprocessor/Microcomputer Terms		44
APPENDIX B: Integrated-Circuit Technologies		50

MICROPROCESSOR APPLICATIONS IN AIRBORNE FLIGHT-TEST INSTRUMENTATION

by
Michael J. Prickett
Naval Ocean Systems Center
San Diego, CA 92152-5000

SUMMARY

This AGARDograph addresses flight-test engineers and flight-test instrumentation engineers interested in the design of microprocessors into new airborne flight-test equipment.

The author has met with several engineers who are actively participating in aircraft- and electronic-flight testing at various organizations. Each organization has developed microprocessor-based instrumentation to solve its unique requirements.

This AGARDograph describes general microprocessor-based, system-design principles and selected, current microprocessor applications for flight test and flight-test instrumentation. It is hoped the reader will gain some new insights to apply microprocessors to future flight-test systems.

1. INTRODUCTION

Since its introduction in 1971, the microprocessor has been used in a variety of ways to make electronic systems more cost efficient (e.g., programmable calculators, electronic games, automotive controls, small general-purpose computers, and flight test instrumentation). Because of standardization, the microprocessor integrated circuit can be made inexpensively, and its architecture makes it very flexible because it functions under program control. Integrated-circuit techniques allow several thousand equivalent transistors to be put onto one chip of semiconductor material, such as silicon or gallium arsenide, in high volume at low cost per chip. Silicon is the common substrate material used at present, while the gallium-arsenide-semiconductor technology promises a much higher functional speed for future integrated circuits. The dual feature of low cost with high flexibility has accelerated microprocessor development and applications.

A microprocessor is a small, monolithic-integrated circuit (often called a "chip") that processes digital data under control of a program. In all modern personal and home computers, the central processing unit (CPU) is a microprocessor that is connected to a number of other units (such as memory, a disc storage system, a printer, and a keyboard) for use as a general purpose computer. Microprocessors are, however, also used to execute specific functions within all kinds of digital data systems. Microprocessors are then equipped with only the memory and the input and output functions required for that one task and are integrated in the system.

Instrumentation systems that include microprocessors are used in both ground equipment and aboard test aircraft to support flight-test programs. This AGARDograph focuses on airborne microprocessor-based systems rather than ground-support systems, and it is these applications of microprocessors in flight-test instrumentation systems that are discussed. Microprocessors are increasingly used in the onboard data acquisition systems and in the associated test and display systems. In Section 2, a general discussion is given of how microprocessors can be effectively used in flight-test instrumentation system applications. That discussion is based on the block diagram of a typical flight-test data acquisition system. Section 3 address ; critical system design factors and the tradeoff considerations that must be resolved during the design process. Section 4 then gives more detailed descriptions of a number of typical flight-test applications. Section 5 concludes the volume with a compendium of commercial microprocessors and support devices.

1.1 BRIEF HISTORY

The first monolithic-integrated circuit, a phase-shift oscillator, was developed in 1958, and the first modest integrated circuit became commercially available in the early 1960s. (Ref. 1). By the early 1970s, several manufacturers were producing various types of digital integrated circuits. As more functions were designed into the chips, the development costs to produce the initial chips became relatively high.

It was thought that if a general digital-integrated circuit could be designed so that it could be programmed by the user for a specific set of requirements, then an architecture or a set of architectures could be standardized, and the cost of the chip design and development could be amortized over many more units, thereby reducing the cost of each unit. Using this principle, Intel introduced the first commercial microprocessor, the 4004, in 1971. The Intel 4040 and 8008 soon followed and in 1974, the 8080. By 1976, several manufacturers were actively supplying microprocessors with large total volumes, causing the unit cost to be significantly reduced.

The major cost reduction of microprocessors caused a greater volume of chips to be used. By 1980, a new, more powerful generation, the 16-bit microprocessor, was being produced: Texas Instruments 9900, Intel 8086, Zilog Z8000, and the Motorola MC68000. The 16 bit refers to the basic word length of the microprocessor. The performance and functions of these processors are discussed in Section 3.2, but in general, these 16-bit processors are faster and perform more functions relative to the earlier 8-bit processors.

By 1984, 32-bit microprocessors were introduced with more functions put on the chip. The Motorola MC 68020 is an example of the family of 32-bit microprocessors, with emphasis on higher execution speed, larger address space (4-Gbyte), and more instructions relative to 16-bit MC 68000 chip. The MC 68020 microprocessor uses the equivalent of approximately 200,000 transistors on an integrated-circuit chip that is approximately 9.5 mm by 8.9 mm. (Ref. 2).

1.2 FUTURE POSSIBILITIES

The future holds interesting possibilities for the tasks that microprocessors will perform for flight-test instrumentation. (Refs. 3, 4). As more powerful processors are introduced into airborne applications, greater amounts of flight-test data can be processed to support the required tasks of the flight-test engineer. His decisions based on this reduced data can be made almost instantaneously so that each flight hour becomes more effective, and the total flight hours for a given flight-test program can be reduced. With the promise of high performance, more low-cost microprocessors are being put into airborne instrumentation systems each year. These microprocessors can be networked to process data from several different systems on the aircraft, making the required data available to the flight-test engineer as the tests are in progress. Future flight testing will become much more comprehensive and cost effective as these trends continue.

The trend for microprocessors and for integrated circuit chips in general is to double the chip complexity each year. This trend has existed since the mid 1960s and will probably continue for several more years, but at a slightly slower pace. (Refs. 5, 6). The implication of this complexity growth per chip to airborne flight-test instrumentation is that measurement equipment will be more comprehensive, smaller, and energy efficient. The functional-performance-per-unit airborne volume is already the limiting factor for many applications. A given airborne-instrumentation system of today could be reduced to a fraction of the present volume, weight, and power requirements and still maintain the required functional performance in the future.

2. GENERAL MICROPROCESSOR APPLICATIONS IN FLIGHT-TEST INSTRUMENTATION

Airborne microprocessor-based instrumentation systems for flight testing can be used where flexible, real-time processing is important. These systems can be grouped into four general elements:

1. airborne sensors;
2. microprocessors and related devices used to control, condition, and store data;
3. data and system status display; and
4. system software.

The block diagram in Figure 1 shows the points where microprocessors are often used in flight-test data systems. This does not mean that they must be used at all those points. In fact, most of the microprocessor functions shown in Figure 1 are often still executed by hard-wired electronic circuits or electro-mechanical devices. The advantage of using microprocessors is the versatility they provide. One microprocessor may use several different programs that can be switched during flight (e.g., a microprocessor can be used to select different sets of parameters from a digital data stream, depending on the type of test executed).

2.1 BRIEF APPLICATIONS

Flight-test microprocessor tasks are now being performed in a variety of aircraft: small aircraft (DEHAVILLAND - 8) (Ref. 7); fighter aircraft (F-15 and F-18) (Ref. 8); and large aircraft (C-141 and commercial airliners); (Ref. 9). The following are examples of typical airborne microprocessor-based projects and the microprocessor that the system uses. More detailed discussions of other selected applications are presented in Section 4.

- The KC-135 Winglets Program. The Intel 8085 is used for system control, data entry, and real-time display of critical flight parameters.
- The F-15 Engine Program. The Intel 8085 is used to interface F-15 engine data to a PCM-serial data link.
- AFTI F-111 Program. The Intel 8085 is used to interface the flight control system to a PCM serial data link.

A series of recently conducted X-29 flight tests illustrates the utility of the microprocessor. A system based on the Zilog Z-80 microprocessor was used to interleave five channels of flight-test data into a single data stream for PCM telemetry transmission. This single data stream was transmitted in real-time for detailed analysis via satellite to Grumman Aerospace Corporation, NY, from the test site at Edwards Air Force Base, CA. The size and functional capability of the microprocessor-based system made this feasible.

The applications of microprocessors indicated in Figure 1, a Microprocessor-based Aircraft Flight-Test System, will be briefly described. It should be stressed, however, that the use of microprocessors is not restricted to parts of such large and complicated flight-test systems. Each of the blocks in Figure 1 can also be applied separately in smaller systems that are used to resolve problems of operational aircraft. The many applications of microprocessors in operational systems of aircraft and in flight-test equipment purchased as a unit (e.g., CRT display units, on-board printers) are not discussed here.

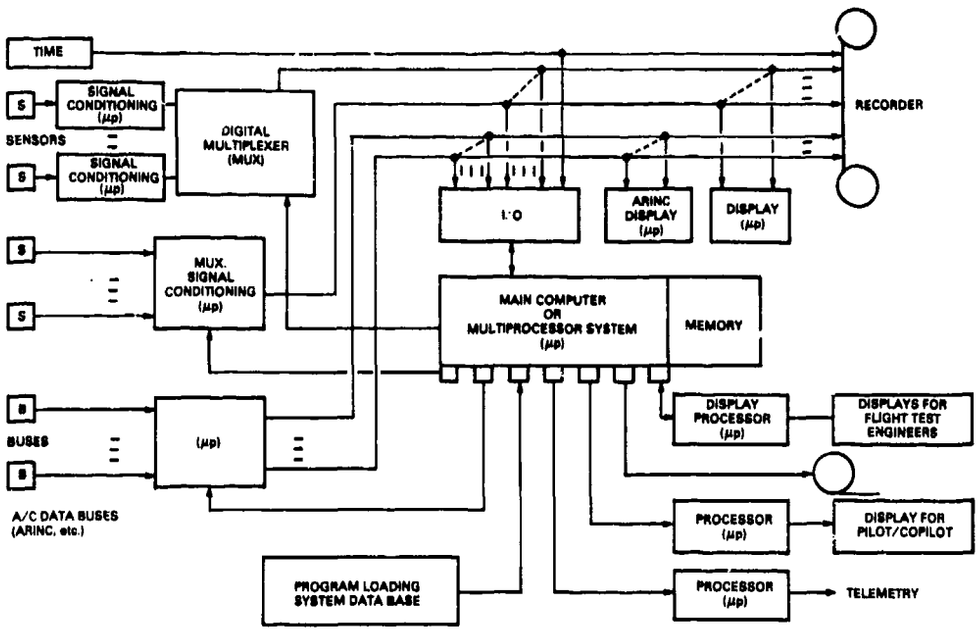


Figure 1. Microprocessor-based Aircraft Flight-Test System.

2.2 SENSOR AND BUS CONDITIONING

A general flight-test application requires that various sensors/transducers be installed throughout the test aircraft. These sensors/tranducers convert mechanical parameters (e.g., pressure, liquid flow, position, acceleration) to proportional electrical voltages and currents. (See Figure 1.) Sensors, S, produce output voltages and currents that typically must be conditioned (i.e., scaled, averaged, calibrated). Signal conditioning can be accomplished using both digital and analog techniques. (Ref. 10).

The signal-conditioning functions may be required before or after the sensor signal conversion to a digital format. System flexibility is substantially enhanced when a microprocessor controls a signal-conditioning process such as voltage scaling or calibration. A single microprocessor can be dedicated to a single sensor or can serve several sensors as shown in Figure 1. For reasons of bandwidth and effective hardware usage, practically all digital flight test systems use time division multiplexing to share a data stream and/or signal conditioners with several transducers (either digital or analog). This process can be done (and has been done for many years) by a hardwired program actuating the multiplexing switches at fixed times determined from a time base. The big advantage of microprocessor multiplexers is that their sampling rate and the selection of the data can be adjusted even during flight. A digital multiplexer under the control of a microprocessor brings the multiplexing process under software control, which makes format revisions much easier. The sensor conditioning and multiplexing functions may each require a dedicated microprocessor or, depending on the requirement, may be shared within a single unit.

In older data acquisition systems all signals were, directly or after preliminary signal conditioning, transmitted to one central multiplexer. This required a large amount of wiring and, especially when the signals were analog, serious danger of electric or electro-magnetic interference. In modern systems, local digitizers and multiplexers produce digital data streams locally. These can be integrated into larger data streams, so that only a few data streams with many channels of multiplexed data arrive at the tape recorder, telemetry transmitter, or central onboard data processor. It is often convenient to use microprocessors for selecting data from data streams. There are two important applications:

- forming configured data streams, i.e., selecting specific data parameters from an existing digital data stream and shaping a new data stream with only those data. This is used in systems where the main data storage is on an onboard magnetic tape recorder, but where selected signals must be telemetered to the ground for reasons of safety and/or optimal conduct of the flight tests. The microprocessor will then select specific parameters from the main data stream (leaving that intact) and produce a new data stream formatted for the telemetry transmitter.
- extracting specific parameters from a data stream that must be further processed for on-line presentation on a CRT or onboard printer or for on-line calculations.

When data from a transducer is in a digital format or when the analog transducer outputs have been digitized, microprocessors can be used for signal conditioning the transducer output; the most important applications are

- (digital) filtering and sampling rate reduction (averaging of several samples measured at a high sampling rate into lower-rate samples);
- linearizing the calibration of transducer outputs, matching of the outputs of several transducers of the same type to the same calibration characteristics; and
- transforming the digital output to engineering units.

Each of these three functions can be used separately or in combination. They are used to speed up the data processing on the ground but are especially important when onboard real-time information is required for displays for the pilots or onboard engineers.

The microprocessor-based system can be networked to other systems via a bus (e.g., MIL-STD-1553B or ARINC-429). A microprocessor can condition or buffer the data from the bus to make the data compatible with the flight-test instrumentation system. Flight-test engineers at Boeing Military Airplane Co., U.S. Naval Air Test Center, MD, and at other organizations have interfaced their microprocessor-based flight-test systems to the existing MIL-STD-1553B data bus in the aircraft. (Ref. 11). The ARINC-429 aircraft data bus is another source of flight-test parameters. Instrumentation engineers at NLR, the Netherlands, and DFVLR, West Germany, have developed microprocessor-based systems to use existing flight-related data on the ARINC-429 bus. These two systems are discussed in Section 4. By using the existing digital-formatted data, hundreds of flight-test parameters are monitored and rapidly processed using a microprocessor.

2.3 FLIGHT-TEST SYSTEM AND DISPLAY CONTROL

The microprocessor is very useful for controlling and performing the various processes necessary for a flight data-monitoring system. The microprocessor can read the conditioned sensor data and aircraft bus data in the proper order. After the data is read from a number of inputs, calculations are performed in real time involving only one or several individual parameters. Examples are the calculation of Mach number from measured static and total pressures, the calculation of the center of gravity of the aircraft from measured fuel flows from tanks located at different places in the aircraft, the calculation of stall speed, and the presentation of pressure distributions in engineering units over wing sections from measured gage pressures. All these calculated values can be presented to the pilots and engineers onboard the aircraft in real time on-line on a CRT or on an onboard printer. These displays can be switched from one set of parameters to other sets, whereby each display can be laid out in an ergonomically optimal way. Another important application of this type is to provide warnings when selected signals exceed or fall short from predetermined values. Examples of parameters suitable for warning monitoring include turbine exhaust temperature, stall speed, and center of gravity. This information is often critical to the pilot or flight-test engineer during real-time flight testing. Flight safety is substantially improved with critical parameters available in real time.

The microprocessor provides calculated qualities for display in different formats, based on the requirements of the test program. By using a set of basic microprocessor instructions, complex tasks can be defined and software developed to fulfill the application requirements. Software development is a significant part of the total system development and should receive no less attention than the hardware. Software development is further discussed in Section 3.

An example of a microprocessor display is an F-14 cockpit unit developed and used by the Grumman Aerospace Corporation, Calverton, N.Y. The Microprocessor Crew Display System (MCDS) was developed to process and display multiple parameters and to allow the pilot to monitor the selected parameters while maneuvering the test aircraft. A critical parameter for maneuvering flight is the "g" level to which the airframe is being subjected. It was learned that normal cockpit "g" meters are not adequate for some dynamic flight-test maneuvers. Certain structural flight-test maneuvers performed by the pilot caused "g" levels to be exceeded by significant margins. This problem was resolved by using the microprocessor-based MCDS together with special accelerometer sensors to monitor, process, and display the aircraft "g" forces in real-time to the pilot. The high-resolution display includes an amber caution light that indicates when a pilot is close to the flight-test envelope. The caution-light threshold is selected based on the maneuver. (Ref. 12).

2.4 SENSOR CALIBRATION

A number of flight-test organizations use microprocessor signal conditioning to calibrate a sensor or group of sensors without removing them from the instrumentation system (sometimes termed "in-situ" calibration). Two general methods accomplish the calibration, one uses a *priori* information about the sensor and the other develops the calibration data during the test sequence.

The first method compensates the offset and nonlinearity of various sensors with a table look-up approach. The table is implemented with a read-only memory (ROM) and controlled by a microprocessor. See Figure 2. The data table with the ROM is developed for the unique output characteristics of each sensor (obtained from the manufacturer or by testing the sensor). Once the ROM is programmed, the microprocessor compensates the sensor's output based on the initial sensor characteristics. This is the static calibration method, since the compensation data is constant with time.

However, the sensor-output characteristics may change as a function of time and environmental conditions. This requires the second method, dynamic sensor calibration, a process performed frequently during the test process with the sensor in place within the instrumentation system. During the test process, the

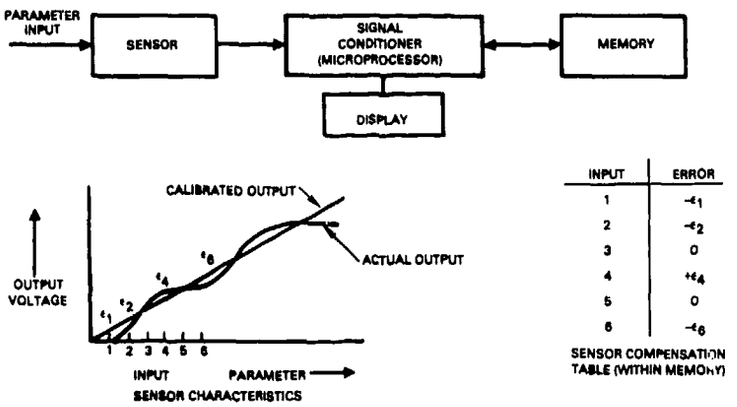


Figure 2. Sensor calibration with microprocessor.

signal-conditioning microprocessor is programmed to initiate known inputs to the sensor at various amplitude levels. The sensor's response to each of these known inputs is measured and stored by the microprocessor over the entire parameter-amplitude range. The difference between the known input and the output response is the error data. The error data may be changed and rewritten into memory using an electrically erasable programmable ROM (EEPROM). This dynamic calibration routine can be programmed into the microprocessor memory to be transparent to flight-test operations. This latter method of calibration is substantially more involved than the first method, but it is worthwhile when higher quality data is required.

2.5 MICROPROCESSOR BENEFITS TO FLIGHT-TEST SYSTEMS

The previous examples illustrate the concrete benefits realized when microprocessors are applied to flight-test systems. Many of the microprocessor functions just listed can be supplied by hard-wired electronic circuits or by electro-mechanical means, and these solutions have worked well in the past. The great advantage of using microprocessors is the great versatility and, in many cases, the higher reliability that is possible. As microprocessors and their associated chips are very small and have a very low power consumption, they need not be mounted in special racks but can be positioned anywhere in the aircraft. The functions they execute can be changed by software. In many cases, several programs can be stored in the memory and transitions from one to another can be initiated during flight from an onboard computer or manually from a keyboard. The possibilities of real-time calculation and calibration of parameters and of preparing good layouts have enormously increased the usefulness of onboard presentation of data and of automatic preflight and postflight calibration and testing.

Where there is no central data processor onboard the aircraft, the functions provided through using microprocessors are still available. The development of very powerful microprocessors during the last decade has made them usable as the CPU of even very large and complex data acquisition systems. Such a central computer can be programmed to execute most of the functions mentioned or can be a central coordinator that actuates all microprocessor circuits everywhere in the aircraft. When a central data processor is available, the use of separate microprocessors will reduce the work that has to be done by the central data processor and significantly reduce the programming complexity.

Microprocessors in flight-test systems increase system flexibility (through software control of the system configuration) and provide rapid assessment of flight-test conditions (through real-time computation of derived parameters). Chosen correctly, such parameters provide substantial improvement in flight safety and data quality.

3. MICROPROCESSOR-BASED, SYSTEM-DESIGN CONSIDERATIONS

The microprocessor is a device that can perform a variety of logical functions specified by a set of stored, programmed instructions. The microprocessor can function as a CPU within a small computer, or as an imbedded processor in an airborne, flight-test instrumentation system. Typical flight-test microprocessor applications include controlling a data-logging process, driving a display, converting data format for a set of flight parameters, and controlling the aircraft control surfaces for special flight testing. Specific flight-test applications will be discussed in Section 4.

The microprocessor should not be confused with a microcomputer. The microcomputer is a composite of a microprocessor, memory devices, special data controllers, and input/output (I/O) interface devices. The microcomputer generally requires several integrated-circuit devices on one or more circuit boards. There are some one-chip integrated circuits that include a microprocessor, limited memory, and I/O buffering functions that together are called a microcomputer chip. This chip is really a specialized controller rather than a computer. The microprocessor chip usually operates with other special-purpose integrated circuits in the chip set. The combined effect is to function as a microcomputer or microcontroller with much more flexibility than a single microcomputer chip. Ordinarily, the microprocessor is fabricated as a single integrated-circuit device.

These are four basic advantages of a microprocessor-based system relati o hardwired, digital-logic designs:

1. Reduction of the number of integrated-circuit devices; this usually causes a reduction in total costs as well as a reduction in system volume, weight, and required power.

2. Arithmetic or computational capability; this is a normal feature of microprocessors.

3. Achievement of "intelligent" operation such that a given process can be adapted depending on external conditions and a stored algorithm.

4. Networking with other microprocessors or computers for distributed data processing.

Each of these four advantages has positive implications for both flight-test and flight-test-instrumentation engineers.

3.1 MICROPROCESSOR SYSTEM ARCHITECTURE

To provide a context for the discussion of microprocessor architecture, consider a general purpose, computer functional diagram, Figure 3.

Figure 3 is a basic configuration for bus-oriented, digital computer systems. The central unit controls the flow of information between the unit, itself and the memory or I/O equipment. This central unit is called the CPU in a mainframe or minicomputer, and it is called the microprocessing unit (MPU) in a microcomputer or in a microprocessor-based system.

The CPU/MPU controls the operation with the memory and the I/O through control lines attached to each unit. These control lines transfer information and comprise the computer system's control bus. In addition to the control bus, there are several parallel lines that convey the information to and from memory and I/O. These lines are grouped together and are called the data bus. A third bus, the address bus,/ points to specific memory locations or I/O devices. In general, digital computer and microprocessor-based systems use the data, address, and control buses to transfer data between the CPU/MPU, memory, and I/O functions. The data and control buses transfer information in both directions and are therefore called bidirectional buses, and the address bus is normally unidirectional.

The four main functions shared with the MPU and the CPU are data transfer, logical, arithmetic, and decision-making operations. The main performance differences between an MPU and a CPU are speed, word size, and total-addressing capabilities. This performance gap will decrease as microprocessor technology advances.

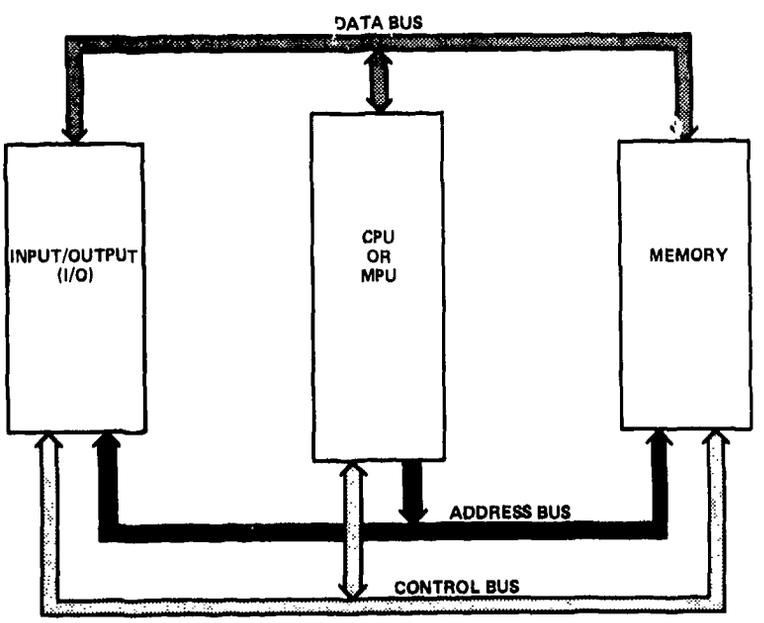


Figure 3. Bus-oriented digital computer.

The internal architecture of a microprocessor, shown in Figure 4, is composed of an instruction register, an arithmetic and logic unit (ALU), a set of registers, and a control circuit that coordinates the operation of the microprocessor. The set of registers includes both special-purpose and general-purpose registers.

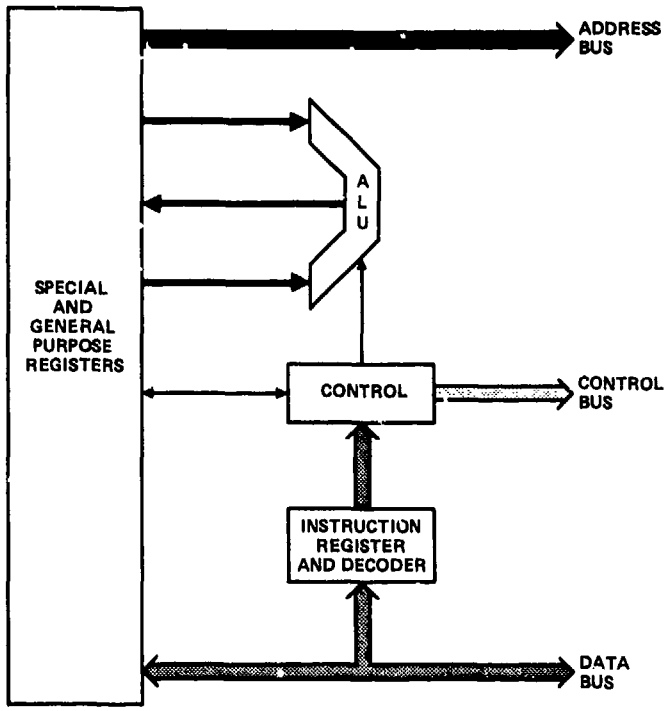


Figure 4. Internal model of a microprocessor.

The control logic causes the microprocessor to perform its two main functions: fetch (or acquisition) and execution. During the fetch phase, the microprocessor receives the next instruction to be executed from memory. Data is fetched into an internal register, called an instruction register, which holds the instruction while the control logic decodes it and begins executing it. During the fetch sequence, an internal register, called the program counter, is incremented enabling the microprocessor to fetch the next sequential instruction from memory. This is the basic sequence required to fetch an instruction and begin executing it. The MPU then begins another instruction cycle.

The ALU's function is to perform the arithmetic and logic operations inside the microprocessor. The control logic directs the operation of the ALU and causes it to perform an arithmetic or logic operation on data from the memory or from an array of registers. The result of this operation is transferred to the memory or to a register to complete the operation.

The general-purpose and special-purpose registers are used for temporary information storage. Typically, a microprocessor has several general-purpose registers that application algorithms can use during the program execution. Special-purpose registers are registers that are optimized for unique functions. Examples of special-purpose registers include an accumulator that holds the results from the ALU, a program counter that contains the address of the next instruction to be executed, and a status register for storage of flag bits.

Since the microprocessor spends much of its time transferring data, this is an important function. Data transfers can follow different paths in most microprocessors. The most common path is to and from memory and I/O.

Microprocessors can also perform some basic logic operations. These operations often include logical multiplication (AND), logical addition (INCLUSIVE-OR), and various forms of shifting and rotating.

Decision-making functions of the microprocessor enable it to control the execution of various selective parts of the program and thereby perform as a controller. All of the decisions that a microprocessor is capable of performing are based upon numerical tests. For example, a number can be tested and the result can indicate a negative quantity. The microprocessor can make a decision based upon this negative result by modifying its instruction flow. Instruction-flow modification is accomplished by a form of conditional-branch or conditional-jump instruction. Other commonly testable conditions are positive, zero, not zero, carry after an addition, borrow after a subtraction, parity even, parity odd, overflow, and equality.

3.1.1 Memory

The memory in a microprocessor system stores the data and instructions of the program. Instructions are stored in the system's memory for quick access. Once the instructions are entered into memory, the microprocessor can execute them at a rate proportional to the clock frequency of the system. It can also use the same sequence of instructions with many different sets of data. This stored-program concept gives the microprocessor its formidable processing power.

In microprocessor-based systems, the program and system diagnostics are customarily stored in a semiconductor read-only memory (ROM). Various forms of ROMs have been developed over the years. All the ROMs mentioned here are nonvolatile and retain data during power interruptions.

- Masked ROM must be programmed while it is being manufactured;
- Programmable ROM (PROM) is user-programmed by burning open, fusible links within the PROM;
- Erasable, Programmable ROM (EPROM) is programmed electrically and erased with ultraviolet light, using EPROM-programmer equipment. This is done while the device is out of the system.
- Electrically Erasable PROM (EEPROM) is programmed and erased electrically, within the system and without special equipment.
- Electrically Alterable ROM (EAROM) has the same basic function as the EEPROM device.

In most systems the ROM stores the program and diagnostics. The PROM or EPROM is used to develop the prototype system. The general function of the EEPROM is to store critical data or programmed instructions for extended periods of time. EEPROM data storage is reprogrammable by the users of the system, but at a slower rate than for reading. The reprogramming feature, while in the circuit, makes it extremely useful for storing aircraft flight parameters and unit conversion tables.

Data is commonly stored in volatile, semiconductor read/write, random access memory (RAM). Read/write memory is available in two basic forms: the dynamic form (DRAM), which requires periodic refreshing; and the static form (SRAM), which retains data as long as power is applied. Because the cheaper, dynamic-memory chip requires additional circuitry to accomplish refreshing, it is normally used in larger memory systems. The term "RAM" has become associated with read/write memory, but read-only memory is also randomly accessible. The various types of ROM (masked ROM, PROM, EPROM, EAROM, and EEPROM) are randomly accessible (as is read/write memory) but are not commonly called RAM. Both ROM and RAM are fabricated from either bipolar transistors or metal-oxide semiconductor field-effect transistors (MOSFETs). The most common transistor type is the complementary, metal-oxide semiconductor (CMOS) and the N-channel, metal-oxide semiconductor (NMOS) memory, which can generally access data in 200 nanoseconds or less.

The system designer must determine the type and relative allocation of read-only and read/write memory appropriate for the application. Most flight-test systems would require both types. Because read/write memory is volatile, provision must be made for power-down time periods. The critical data that cannot be lost should be stored in either EEPROM, or battery back-up read/write devices, or a combination of both.

3.1.2 Special Purpose Registers

The Accumulator: Generally, the accumulator is the principal user register in the microprocessor. In many microprocessors, the results of either the arithmetical or logical operations performed by the ALU are transferred to and stored in the accumulator.

The accumulator is found in most microprocessors. It stores one of the operands to be used by the ALU in performing arithmetical or logical operations. The accumulator can function both as a source register and as a destination register. As a source register, the accumulator can be programmed to add its own contents with the contents of a memory location and to store the result in the same or another memory location. The accumulator becomes both a source and a destination register when the result is stored in the same accumulator.

Typically, the accumulator and general-purpose registers in the microprocessor are designed to perform other functions such as complementing the contents, shifting the data right or left, and rotating the data right or left. Also, the accumulator is often designed to store a cumulative or progressive total of the numbers transferred into it. Each successive operand or number transferred into it is added to the previous sum.

The Instruction Register and Decoder: The microprocessor's word-length-in-bits is established by the size of the internal storage registers (counters, accumulators, and internal data transmission buses). A word is the basic unit of bits that the machine handles as a group. Most one-chip microprocessors use 8-, 16- or 32-bit word lengths. In microprocessor terms, a 4-bit field is called a nibble and an 8-bit field is a byte. Each operation performed by the MPU is identified by a unique group of bits called instruction codes. When an 8-bit instruction is used, it is possible for that system to have up to 256 unique, single-word-operational instruction codes ($2^8 = 256$).

Fetching the instruction from the program memory involves two separate operations. First, the MPU transmits to the memory the address of the instruction in the program counter. The memory then transmits the contents of the addressed location to the MPU, where it is temporarily stored in a dedicated register called the instruction register. The contents of the instruction register are then decoded by the decoder and, in association with timed clock pulses, the appropriate data transfer paths within the system are established, and the various activities called out by that instruction are executed.

Although 8 bits may be adequate for instruction codes in some microprocessors, there are cases where more than 8 bits may be required (e.g., an instruction that references a fairly large data memory). Here, the 8-bit (1 byte) instruction can identify the operation to be performed but not the operand location. A 2-byte or even larger instruction is necessary to specify the data location.. Such multibyte instructions are stored in adjacent memory locations. The MPU then performs two or three fetches in succession, as appropriate, to acquire the full instruction. Where multibyte instructions are involved, the first byte, which usually contains the operation code of the instruction, is transferred into the instruction register. The remaining byte or bytes are placed in temporary or auxiliary registers.

The Address Register/Counter: The address register is a temporary storage device for holding the location address to be accessed for input/output of data in data memory. This register specifies what is put on the address bus. The number of address bus lines specifies the number of unique address locations that can be specified. For example, the 8-bit MC 6800 microprocessor has 16 address lines. Thus, its address space is 65,536 (2^{16}). The MC 68020 can directly address 4,294,967,296 locations with a 32-bit address bus (2^{32}).

The Program Counter: Program instructions are stored in consecutive locations within the program memory, which can consist of several memory chips. Each memory location has a unique address. In order to execute the program in the properly assigned sequence, the microprocessor must know where to go in the program memory to fetch the instruction. This is done by the program counter, which contains the address of the next instruction to be fetched. The microprocessor updates or increments the program counter every time it fetches an instruction.

3.1.3 Program Execution Control

When a JUMP instruction is encountered in the program, the normal sequence is suspended. The program counter is directed to the address specified by the JUMP instruction rather than to the next sequential address. This provides logical continuity to the program. A JUMP can be either a go forward or a go backward.

Subroutines are programs within a program. During a program run, a certain group of instructions are often used over and over again. Repeating these instructions every time they are needed is wasteful of program memory. Therefore, special routines can be written once, stored in the memory as a subroutine, and called out by the main program when needed. BRANCH is a type of JUMP instruction that can also be used to call a subroutine. The JUMP/BRANCH instruction contains the starting address of the subroutine which is automatically inserted in the program counter.

To ensure an orderly return to the main program after completion of the subroutine, the address of the next sequential instruction must be in the main program, following the JUMP/BRANCH instruction to be stored. Prior to the branching and execution of the subroutine, the microprocessor increments the program counter and stores this address in a memory area called the stack. A stack is a dedicated memory area that may be part of the normal read/write system memory.

The last instruction in the subroutine is a BRANCH BACK or RETURN instruction that returns control back to the main program. The microprocessor inserts the address at the top of the stack into the program counter, and the main program is resumed at the address immediately following the JUMP/BRANCH instruction.

Nesting is the process by which one subroutine can call out a second subroutine that, in turn, may call out a third subroutine, and so on. The number of subroutines that can be nested depends on the number of return addresses that can be saved by the microprocessor. In other words, the depth of nesting is determined by the depth of the stack.

3.1.4 The Stack

The stack is usually a section of read/write memory where the contents of the memory are accessed on a last-in/first-out (LIFO) basis. When a JUMP/BRANCH instruction is executed, the following events generally take place.

1. The contents of the program counter are incremented.
2. The contents of the program counter and other registers are transferred to the stack.
3. The starting address of the routine, which is contained in the JUMP/BRANCH instruction itself, is transferred to the program counter.
4. The routine is executed.
5. After completion of the routine, control is returned to the main program by transferring the contents of the stack back into the program counter and other appropriate registers. The microprocessor is then in the same status as just before the interruption.
6. The main program resumes its normal operation.

3.1.5 Instruction Cycle

The microprocessor performs only what it is instructed to do. Generally, several activities must take place in the microprocessor for each instruction in the program. The instruction cycle covers a certain time span. Typically, it takes several clock cycles to make one instruction cycle. The ratio between clock and instruction cycles is a function of the microprocessor design and the instruction being used. Figures 5 and 6 illustrate microprocessor instruction cycles.

3.1.6 Interrupts

The interrupt inputs on most microprocessors can, when placed at their active level, interrupt the program and transfer control to a subroutine from the memory. The purpose of this subroutine, called an interrupt-service subroutine, is to service the interrupt. Interrupts handle extremely slow external I/O devices, such as keyboards. Rather than requiring the microprocessor to wait for the input of slow data or to constantly check on the status of input data devices, interrupts allow normal execution of instructions until I/O devices or special events need attention.

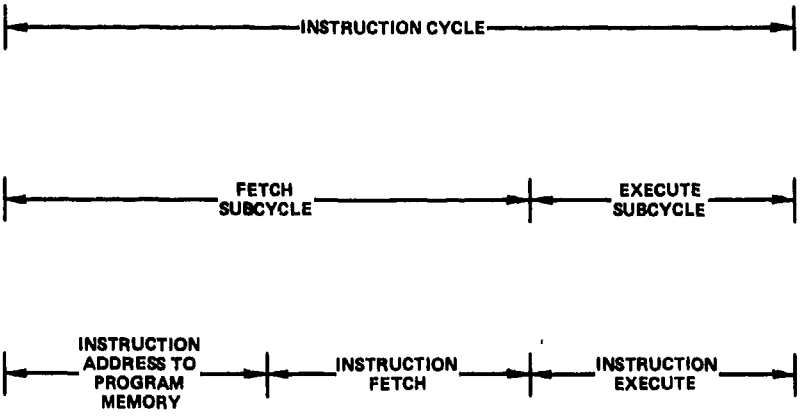


Figure 5. The typical instruction cycle for a microprocessor.

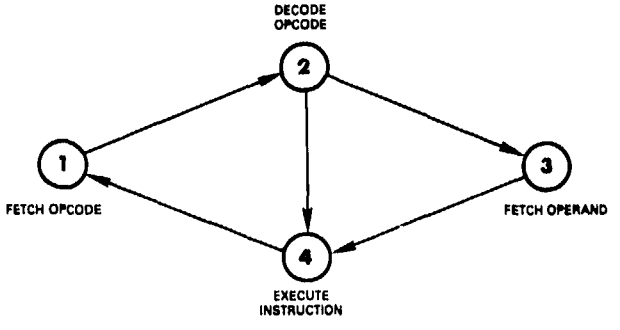


Figure 6. Machine instruction cycle (Van Neuman cycle).

Interrupts can be simple or complex depending on the design of the microprocessor. Typically, a microprocessor has different levels of interrupt priority that can be handled by a "vectored-interrupt" technique. Various interrupt sources can cause the MPU to react to each with an appropriate response. There can be both software interrupts (JUMP to SUBROUTINE) and hardware interrupts (from peripherals).

A priority-interrupt system can be very useful in airborne flight-test situations. For example, in a flight-test, data-logging system, if system power voltage is being lost, the proper interrupt can cause the microprocessor to store all the critical flight-test data to some non-volatile memory before the minimum voltage is reached.

3.1.7 The Microprocessor Chip Set

The microprocessor-chip set is the set of devices that function under the direct control of the microprocessor to achieve the designed purpose. Normally these devices are connected to the microprocessor via the data, address, and control buses. The following are examples of device functions within the chip set; the microprocessor, read/write memory (RAM), program memory (EPROM), critical flight-test data (EEPROM), serial and parallel data I/O, programmable timer, priority-interrupt controller, floppy-disk controller, arithmetic processor, direct-memory-access (DMA) controller, address decoder, bus buffering, clock generators, etc. Several of these specialized devices are made for most microprocessors. Flight-test-instrumentation engineers have many design options available to develop a microprocessor-based system to achieve their requirements. The availability and cost of these support chips are significant factors in the selection of a given microprocessor.

Figure 7 is an example of a microprocessor-chip set used in a flight-test application. The chip set illustrates a microprocessor, memory, and various special-purpose I/O devices. The function of this system is to continuously monitor several transducers. If the measured parameter of any transducer is outside a preselected range, the transducer label, parameter value, and time can be recorded and transmitted, in real time, to a ground station. The microprocessor can also convert the flight-test parameter into the appropriate engineering units before the information is recorded and transmitted.

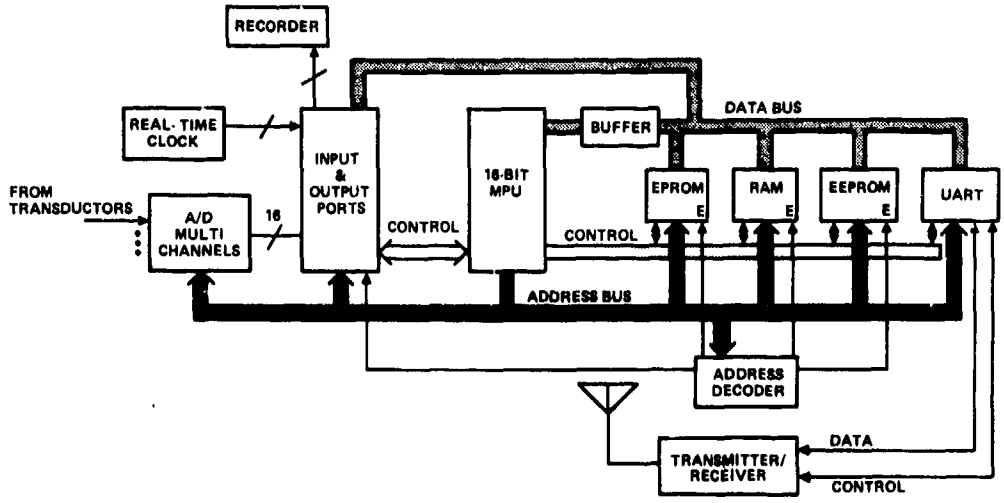


Figure 7. 16-bit microprocessor-chip set.

All the devices shown in Figure 7, except for the transmitter, receiver, recorder, and real-time clock are part of a normal microprocessor-chip set. This system can receive a ground station's coded commands to reprogram through the Universal Asynchronous Receiver-Transmitter (UART). The UART is a special-purpose integrated circuit often used for general-purpose serial communication. UART transfers serial data from the microprocessor data bus to the aircraft PCM transmitter/receiver unit. Because it is bidirectional, the UART can also transfer data to the microprocessor via the data bus. Due to unexpected flight conditions, reprogramming may be necessary after flight operations have started. With the EEPROM and UART in the chip set, program changes can be implemented from a remote location.

There is ongoing research and development of specialized devices for the microprocessor-chip set. For example, a special integrated-circuit device has been developed that is able to code and decode digital serial data at a rate of 680 MBPS (Ref. 13). With continued device development, microprocessors will be able to control high-speed data links and provide more flight-test data at an even faster data rate.

3.1.8 Direct Memory Access (DMA)

High-speed I/O peripheral devices usually exceed the speed capability of the microprocessor. A special device in the chip set, the DMA controller, can be used to transfer large amounts of data at high speed.

DMA is an I/O method in which special hardware performs most I/O operations. The microprocessor CPU turns over control of the buses to the DMA controller, which then transfers data directly between the memory and the I/O section. The DMA controller usually transfers an entire block of data; it provides addresses and control signals to the memory, updates the addresses, counts the numbers of words in the transfer, and signals the end of the operation.

The advantage of DMA is speed. Transfers can proceed at a rate limited only by the access time of the memory. The microprocessor CPU need not fetch and decode the instructions that would transfer the data, nor update the address and the counter, nor check for the end of the operation. Instead, the DMA controller performs these tasks at a speed that far exceeds most programmed data-transfers to or from the microprocessor.

Usually, when hardware such as a DMA-controller device replaces software, there is a tradeoff between the speed of hardware and the greater flexibility of software. DMA controllers are rather complex. The DMA controller is essentially a specialized processor, since it transfers data much like the normal microprocessor.

Many microprocessors have a DMA controller within an available chip set. These are the basic functions of DMA:

1. To bypass the microprocessor during the DMA operation.
2. To control the buses so as not to interfere with normal microprocessor activities or cause bus contention (data conflicts).
3. To determine the length and location of the transfer.
4. To mark the end of the operation.

Another method of faster memory access using two microprocessors is dual-ported memory. Memory devices that are dual-ported increase the effective data rate of memory transfer by allowing connection of the microprocessors to a common-memory device. This has the advantage of two different microprocessors using the same data set at individual rates. The alternative approach is to have one microprocessor control access to the memory device, thereby requiring the other microprocessor to interrupt it for memory access (if only single-port memory is available). Dual-ported memory devices reduce the microprocessor waiting periods and increase effective data-memory transfer. In the rare instance that two microprocessors address a common-memory location instantaneously, the on-chip arbitration system settles the dispute by allowing both to use the common location, but at a slightly different instant in time. The available support chips of the selected microprocessor determine the feasibility of either DMA or dual-ported memory.

3.1.9 Bus Buffering/Demultiplexing

Buffering the microprocessor buses is important. When several different chips are connected to the microprocessor buses in parallel, the equivalent electrical capacitance becomes extremely large. The current-driving capability is increased by buffering the buses to drive this additional capacitance. Generally, if signals must be extended off the microprocessor board, they should be buffered.

Several microprocessors have a bus that multiplexes various types of signals. This class of microprocessors requires buffers to demultiplex the signals and route them to the proper locations. An example is the Intel 8085A microprocessor. The flight-test group at the Aeroplane and Armament Experiment Establishment (AAEE), Boscombe Down, United Kingdom, has used the Intel 8085A to control an in-flight failure-simulation system. Also, NASA Dryden Flight Research Facility at Edwards Air Force Base, California, has used the 8085A to control an aircraft stall-speed warning system. Both these systems are discussed in Section 4.

The Intel 8085A multiplexes the least significant 8-bits of the address lines with the 8-bits of data (AD0-AD7). This is done primarily to reduce the number of pins on the microprocessor chip. A signal is provided to demultiplex the bus, Address Latch Enable (ALE). Figure 8 illustrates how the demultiplexing is accomplished. The microprocessor sends out the least significant half of the 16-bit-memory address on the address/data bus with the ALE signal. An octal transparent latch captures this portion of the address and holds it until the microprocessor changes the memory address along with another ALE. When the input "G" signal is high, the latch accepts data and is transparent to the data, so it appears at the latch's output. When ALE goes to the logical low state, the 8-bits of address are latched or captured. With the latch temporarily frozen, the microprocessor then presents new data to the same microprocessor output pins.

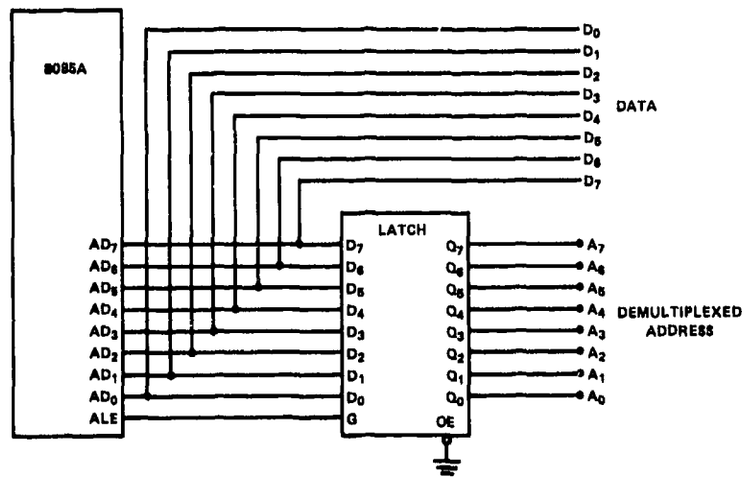


Figure 8. The Intel 8085A microprocessor with address and data demultiplexing.

3.1.10 Three-State Devices

Three-state logic devices help control the flow of information on the buses within the chip set. If the microprocessor and all the various devices within the chip set have three-state (sometimes called "tri-state") output capability, the microprocessor can effectively control data transfer. Two three-state output devices, such as two RAM memories, can have outputs connected to a common bus, if the outputs are not active at the same time. If the microprocessor-based system is properly designed, one memory is active, while the other memory and all other devices on the common-data bus are inactive. Thus, there are no data-state conflicts. The proper control input on a three-state device causes the device to be inactive (output in a high-impedance state), which has the effect of separating the device's output from the common-data bus. There are three output states:

1. logic high;

2. logic low; and

3. high impedance, if the device is inactive.

Bus contention is the term generally used to describe a data conflict when one active device is outputting a logic high, and another active device is simultaneously outputting a logic low on a common-data bus line.

3.1.11 Memory-Mapped I/O and Isolated I/O

Various microprocessors control I/O differently. One of these techniques is memory-mapped I/O, in which I/O devices are treated as memory. (The I/O is processed in the same way as memory with regard to software and hardware control signals.) The other technique is isolated I/O or I/O-mapped I/O, in which the I/O device is treated differently. The I/O device has a unique I/O address, not a memory address. Microprocessors that use this type of I/O have special instructions, IN and OUT, to transfer data to and from this separate I/O space.

Both I/O control techniques have advantages and disadvantages. In the memory-mapped I/O scheme, there are two basic control signals: memory read and memory write. In the isolated I/O scheme, there are four basic control signals: memory read, memory write, I/O read, and I/O write. The main advantages of the memory-mapped I/O system are that it allows more space for I/O devices; it allows more instructions to operate on the I/O; and it requires fewer instructions. The isolated I/O scheme is a simpler design and does not require any memory space for I/O to function. The microprocessor has reserved special space for it. More memory can be used because I/O is not in memory space.

3.1.12 Microprocessor Instructions

The microprocessor program consists of a series of instructions. Each instruction is a command to the microprocessor to perform a certain specified task. The set of instructions available is unique to a microprocessor. When writing a program, the programmer is restricted to only those instructions provided by that particular microprocessor. The program is a function of both the overall task as well as the particular microprocessor instruction set. Microprocessor manufacturers group their products into processor "families." The microprocessors within a given family have similar instruction sets.

Instruction can be categorized in various ways in terms of function. Some instruction categories and examples for each are listed:

1. Data-manipulation instructions transform the data in some way. Such instructions generally use the computer's ALU. Common subcategories include

   a. Arithmetic instructions
   b. Logical instructions
   c. Shift instructions
   d. Comparison instructions
   e. Special-purpose instructions

2. Data-transfer instructions move data from one place in the computer to another area, without actually changing the data. Common subcategories include

   a. Memory-transfer instructions
   b. I/O instructions
   c. Internal-transfer instructions
   d. Stack instructions

3. Program-manipulation instructions transfer program control from one place in memory to another. They change the program counter so that instructions are executed out of their normal sequential order. Common subcategories include

   a. Unconditional JUMP instructions
   b. Conditional JUMP/BRANCH instructions
   c. Subroutining instructions
   d. Halt

4. Status-management instructions change the status conditions of the computer without affecting the data or the order in which instructions are executed. These instructions perform management functions rather than data-processing functions.

   a. Transfer accumulator to status-register instruction
   b. Clear interrupt-mask instruction
   c. Set interrupt-mask instruction

Address modes are used with the instructions so that memory or register addresses can be used in an effective manner. The address should be as short as possible. Short addresses take less memory space and less time to fetch. However, a large number of memory spaces can be useful and cannot be specified with a short address. This apparent contradiction is resolved by using one of several address modes, with an instruction to do a particular task. The following example uses the Motorola MC 6809. The "A" accumulator is loaded using four different address modes. The number of bytes required and the microprocessor MPU cycles to fully execute the instruction are given:

1. The Immediate-Addressing Mode: Load "A" with the next immediate byte (the first byte is the operation code; the second byte is the data for "A"). This is a two-byte instruction, and it requires two MPU cycles.

2. The Extended-Addressing Mode: Load "A" with the contents of the address specified by the next two bytes (16-bits). This is a three-byte instruction, and it requires five MPU cycles.

3. The Direct-Addressing Mode: Load "A" with the contents of the address specified by the next byte (this mode is only useful for the very low end of memory). This is a two-byte instruction, and it requires four MPU cycles.

4. The Index-Addressing Mode: Load "A" with the contents of the address specified by the addition of the next byte and the contents of one of the index registers (an internal 16-bit register). This is two-byte instruction, and it requires at least four MPU cycles.

Proper consideration of the instructions as well as the different addressing modes yields the optimum program. Microprocessors generally have from 50 to 200 instructions and 10 to 30 address modes. A computer- or special-software-development system is frequently used to help the programmer. Software development is discussed later in this section.

3.1.13 Multiprocessing or Networking

The concept of either networking several microprocessors into a larger system or connecting systems together is important in some airborne flight-test applications. Section 4 details several applications where the flight-test system is based on distributed microprocessors or where a single microprocessor is connected to a data bus of a different system on the aircraft. These are two purposes of networking microprocessors:

1. To optimize several subordinate processes with individual microprocessors distributed throughout the aircraft. These processes would then be coordinated with a master-control unit, such as the Lockheed Georgia "LADDS" system. (Section 4.1).

2. To provide a single microprocessor-based display system. The system would present various test parameters for monitoring during aircraft flight. It would require an interface to an existing data bus such as the NLR ARINC display system. (Section 4.4).

Networking requires a set of standard-information buses between systems and subsystems onboard the aircraft. Two common standard-data buses for aircraft are the ARINC 429 (for commercial aircraft) and the MIL-STD-1553 for newer US military aircraft. Both these buses contain data that flight-test systems could use. (Refs. 14, 15).

In order to network microprocessors together, a number of buses are defined and used. The choice of microprocessor buses is determined by the microprocessor selected because manufacturers have tried to "standardize" their product devices to a bus.

There are a large number of microprocessor-related buses, each having advantages and disadvantages. (Refs. 16, 17, and 18). The following is a list of the more popular buses that connect support devices to microprocessors. Major sponsors and a brief description of data and memory-address buses are included.

- MULTIBUS I, Intel, 8/16-bit data, 24-bit memory
- MULTIBUS II, Intel, 32-bit data, 32-bit memory
- VME, Motorola, 16/32-bit data, 32-bit memory
- VERSABUS, Motorola, 16-bit data, 24/32-bit memory
- STD Bus, Pro-Log/Mostek, 8-bit data, 16-bit memory
- S-100, IEEE-696, 8/16-bit data, 24-bit memory

3.2 MICROPROCESSOR SELECTION CONSIDERATIONS

Because many microprocessors are available, screening a microprocessor for a required flight application is critical. The following guidelines are intended for use by the flight-test instrumentation engineer when selecting a microprocessor.

3.2.1 Application Considerations

First, determine the general application. How will the microprocessor be used? Will its primary function be as a controller or as a data processor?

A controller may be I/O intensive, require special interfaces, use real-time interrupts, use a relatively small amount of memory, and directly use assembly language for program development.

A data processing application may require a math coprocessor or an enhanced set of arithmetic instructions, several addressing modes, a relatively large address space, and high-level-language software development.

Next consider if the application is dynamic. Is it subject to change? It is the nature of most flight applications to change and expand functionality as the application matures. This tendency should be carefully assessed at the outset of the project, when it is far easier and often more economical to over-design a system in the area of computational capability. In this way, major system expansions can often be addressed without major redesign. This may determine type of memory used, how modular the hardware design, and how flexible the software. A program written in a high-level language with the proper software development equipment can be flexible, but it requires more execution time and memory space.

Also, does the application require a relatively long or short word to represent the data being processed? Should an 8-bit, 16-bit, or 32-bit microprocessor be used? An 8-bit can normally do what a 32-bit MPU can do, but it will take more time. Other considerations include type of number structure (fixed/floating point), computation speed, and specialized data handling.

3.2.2 Implementation Considerations

Microprocessor implementation such as single-chip, multichip, or bit slice (for custom design) is another factor in performance/development cost considerations.

Single-chip microcomputers, available from various manufacturers, are useful in less complex applications. These one-chip microcomputers contain a microprocessor CPU, ROM, EPROM or EEPROM for program storage, and RAM for data storage. In most cases, they also contain I/O operations. Small-sized systems can be implemented by using one of these devices and very few other components. This decreases the expense of development time for both hardware and software. Also, the fewer components in a system, the easier it is for field servicing.

Bit-slice technology is appropriate only when a flight application requires extremely high speed, very wide wordwidths, or a tailored instruction set. Since the instruction set of the bit-slice microprocessor is custom-relative to the fixed-wordwidth microprocessor, more time is needed for software development. A typical fixed-wordwidth microprocessor, such as the Intel 8085A, has a few hundred instructions, whereas the bit-slice microprocessor may contain tens of thousands of custom instructions. Developing a bit-slice system requires significant increases in cost and development lead time.

The multichip-set standard microprocessor is by far the most popular and flexible. It is a good compromise between the limited performance of the single-chip microcomputer and costly bit-slice technology.

3.2.3 Technology Choices

There are two main technologies used to fabricate microprocessor and other chip-set devices: bipolar and metal-oxide semiconductor (MOS). These terms refer to the type of transistor being used within the device. The Bipolar Junction Transistor (BJT) and the MOS transistor are fabricated differently. Common digital-logic device groups with bipolar transistors include transistor-transistor logic (TTL), and emitter-coupled logic (ECL). MOS transistors are used in NMOS and CMOS. The transistor-fabrication methods, as well as the design of these common digital-logic gates, are presented in several textbooks such as those by Millman and Mead. (Refs. 19, 20).

It is difficult to compare MOS and bipolar devices due to ongoing development, but MOS devices tend to be more dense (more transistors/functions per chip) and require less power. However, they are slower than bipolar devices. There are some integrated devices that have both MOS and bipolar transistors on the same chip. There are several variations of TTL, ECL, NMOS, and CMOS devices.

The technology of microprocessor fabrication affects several features of flight-test design such as speed, power consumption, electrical noise margin, and number of integrated-circuit chips within the system. A few years ago, CMOS was commonly considered slower than TTL-digital gates and was used only in low-speed applications. Today, high-speed versions of CMOS are commercially available, and CMOS has many applications, including flight-test instrumentation. Most microprocessors are now made using MOS transistors because of their relatively low power consumption and high density characteristics.

3.2.4 General Selection Considerations

The system designer should also consider these factors when choosing a microprocessor for flight-test instrumentation:

1. Microprocessor architecture; both internal- and external-chip set.
2. Data wordwidth - for resolution of results and system speed.
3. Memory-addressing capability.
4. Chip-set completeness.
5. Multiprocessing/Networking.
6. Instruction set/Addressing modes.
7. Instruction-cycle time - not merely clock rate.
8. Interrupt capability.
9. Hardware and software development aids.
10. Power requirements.
11. Environmental considerations, such as military qualifications.
12. Price and availability of the total chip set.
13. Second and even third sources for the chip set.
14. Policy constraints (e.g., MIL-STD-1750 ISA, "ADA" Programming Language).

These topics should be reviewed together. All considerations are important, but hardware and software development aids should be emphasized. While these development aids represent a large investment, they do make the system development task much more productive.

3.3 SOFTWARE DEVELOPMENT

Software development involves all activities associated with the successful organization and efficient operation of the microprocessor instruction set. Development depends on the microprocessor selected. Once a selection has been made, software development can be based on an existing "development system." Many systems are available to help engineers develop microprocessor-based systems. Flight-test application software can then be efficiently written, object-coded, debugged, executed in both simulation and emulation (real-time) modes, and time analyzed. Several high-level programming languages can be used, such as FORTRAN, BASIC, PASCAL, "C", and FORTH. Most microprocessor manufacturers and some general-instrumentation companies build software-development equipment.

Although software development is a function of the design engineer's specific development aids, some general principles apply. On the average, software (including software maintenance) increases total system costs as more computing systems are developed and used. (Refs. 21, 22).

In regard to software development, it is also important to define the basic distinctions among the three levels of programming languages:

1. High-level language (HLL) is problem-oriented and requires several microprocessor instructions to implement. The HLL program is usually not a function of the target microprocessor.

2. Assembly-level language is specific to the target microprocessor and is represented by mnemonic characters. Most HLL program statements will translate into several assembly statements.

3. Machine-level language instructions have a one-to-one relationship with assembly statements and therefore are specific to the target microprocessor. Machine-level language is represented by binary numbers.

Table 1 indicates the basic differences between working with HLL and assembly programs.

Table 1. Distinctions between high-level and assembly languages for software development.

	High-Level Language	Assembly Language
Coding	Reasonably efficient.	Time consuming.
Testing	Reasonably efficient. Many tests are performed during compiling.	Time consuming. Some tests are performed during assembly.
Execution Speed	Less effective.	More effective.
Memory Space		
Documentation	Efficient. Some HLLs are self-documented.	Great effort.
Maintenance and Modification	Can be done quickly and safely.	Greater effort.
Some Applications	Long general programs.	Drivers. Special high-speed tasks.
Objective	Reduction of overall software-development time.	Speed and less memory.

3.3.1 Software Development Stages

Microprocessor software development can be divided into several stages:

1. Problem Statement and Specification. This is the most important stage in the software-development cycle, and it is the starting point of all succeeding activities. The problem to be solved is defined, the various flight tasks to be performed are clearly outlined, and specifications are written. The specifications should be clear, concise, and written so the user can readily understand them. The various inputs and expected outputs are specified, along with any other constraints or limitations that may be recognized and known. Procedures for handling errors should also be included.

2. Program Design. This stage encompasses the design of a program to meet the requirements of the problem definition. Useful techniques include top-down design, structured programming, modular programming, and flowcharting.

3. Program Coding in HLL. This stage consists of transcribing the previously designed program into an HLL that can be translated into the machine language. The program produced in HLL codes is called the source program. Examples of HLL include FORTRAN, FORTH, ADA, PASCAL, JOVIAL, BASIC, "C", or Programming Language for Microprocessors (PL/M).

4. Translation to Microprocessor. To translate the HLL source program to the microprocessor object code requires a compiler program. The compiler is a program written to be executed on a specific computer, using a specific HLL program as a source (the compiler input) and a specific microprocessor for its object-code results (the compiler output). The program then produced in the microprocessor's machine language is called the object program. If the source program is written in the assembly language of the target microprocessor, then an assembler program is executed to translate the source to the microprocessor object code. If the assembly program is written to be executed on a general-purpose computer, it is called a cross-assembler.

5. Debugging. Debugging (sometimes called program verification) entails the discovery and correction of programming errors. Since few programs run correctly the first time, debugging is an important and time-consuming stage of software development. Useful debugging or verification techniques include editors, debugging packages, software simulations and emulations, logic analyzers, breakpoints, trace routines, memory dumps, and software interrupts.

6. Final Validation or Testing. Program-execution testing confirms all required tasks over all specified conditions.

7. Documentation. Program documentation is necessary for user information, for maintenance, and for future applications. Flowcharts, commented program listings, and memory maps are widely used in program documentation.

8. Maintenance. Maintenance includes updating and correcting the program to allow for changing conditions and field experience. Proper testing and documentation should significantly reduce the frequency and extent of software maintenance.

9. Use and Re-design. This stage of software development extends the program to solve tasks beyond the initial problem definition. Designers should take advantage of programs and equipment developed from previous tasks. Requirements for future tasks should always be considered.

Since each stage of software development affects other stages, problem definition must include consideration of a test plan, documentation standards, maintenance techniques, and the possible extension to other tasks. Inherent in program design are provisions for debugging, testing, and documentation. (Ref. 23). Poor program design causes wasted effort in testing and debugging. The quality of documentation also has an impact on maintenance and redesign.

Determining the availability of an assembler and compilers is crucial to software design. If software tools are available, the application program can be written in FORTRAN or other HLL and then translated to assembly rather than machine language. An experienced programmer can review the assembly statements and make changes to increase speed and decrease program size.

Figure 9 illustrates the interaction in the software development cycle. Initial planning is critical for the development project to stay on schedule. An example of how to conduct an extensive software-development project is discussed by Howes. (Ref. 24).

The structured approach is also emphasized in Refs. 25, 26. If software performance is critical to flight safety, quality assurance must be built in.

Most often, the flight-test system does not have to meet a government specification or general standards. However, a flight-test engineer or instrumentation-design engineer may find such specifications and standards useful in planning for software and hardware. The IEEE Software Standards, the U.S. Government, and RTCA, for example, provide guidelines for a software development program and methods for quality assurance, documentation, and testing. (Refs. 27, 28, 29,).

3.3.2 Hardware and Software Tradeoffs

Microprocessor-based systems in general, and flight-test, microprocessor-based instrumentation in particular, require a high degree of coordination between hardware and software development. The design team or system-design engineer(s) must consider hardware and software interactions before the final system-design decisions are made. The addition of special devices in the microprocessor chip set can greatly relieve the software task (e.g., an arithmetic processor in a flight-test system can do several calculations involving transcendental functions). Such calculations are performed faster and with much shorter programs than if the microprocessor had to do them with its basic instruction set.

Several functions can be accomplished in hardware or software. The functions that are best done in hardware or in software are application dependent. Special devices, circuit design, or software routines within the main program are all valid options.

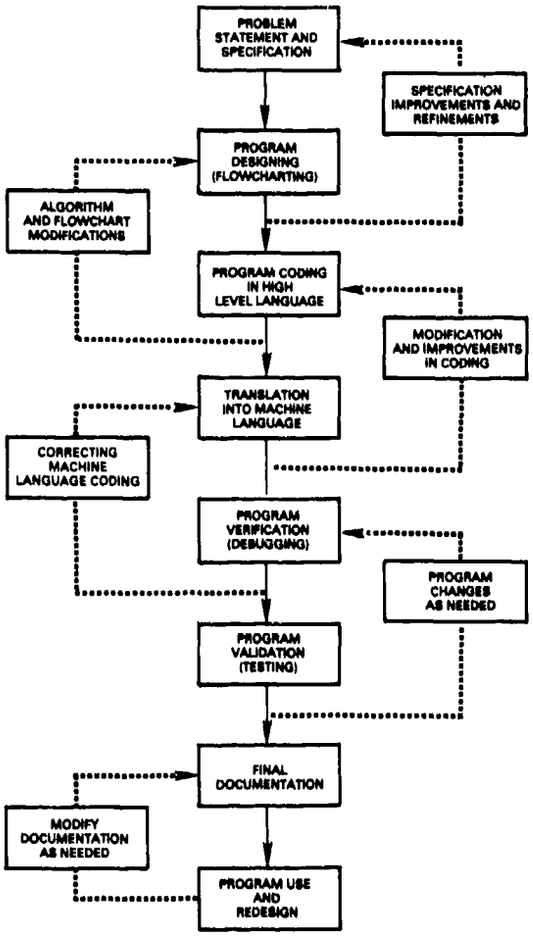


Figure 9. Microprocessor software development cycle.

Nevertheless, there are important tradeoffs within the software development process. The primary programming language has a bearing on several factors of the microprocessor system. Some factors are the memory required, the execution speed, program flexibility, and software documentation. The primary programming language is application-dependent. For short programs where execution speed is critical, the program can be developed in the microprocessor's assembly language. Only a small amount of memory is used, and the project does not require the investment of a development system. This author was involved in developing a 16-bit, 8086 microprocessor-based radar controller in which system speed was critical. With special peripheral circuits, the microprocessor did various calculations based on several input-data channels, and the results were outputted every 20 microseconds. The entire control program was developed in 8086 assembly language and required less than 800 microprocessor words of program memory (1.6K bytes).

If execution speed is not as critical, but still important, and software development aids are available, the system designer should consider the HLL programming languages, FORTH and "C". Both are used for I/O device control where execution speed is a main consideration. Both are used for medium or long programs (relative to assembly language).

In summary, the functions performed in hardware/software and the primary programming language selected are ultimately determined by application, design engineer's experience, and available design-support equipment.

3.4 MICROPROCESSOR SYSTEM DESIGN-SUPPORT EQUIPMENT

The design of both software and hardware for a microprocessor-based flight system depends greatly on the available support equipment. Support equipment includes software programs for various microprocessors, specialized hardware, and general electronic laboratory equipment.

The following are categories of design-support equipment:

1. Logic analyzers
2. Emulators
3. PROM/EPROM programming equipment
4. General computers (with appropriate software)
5. Microprocessor development systems (many include the previous four categories)

3.4.1 The Logic Analyzer

While the triggered oscilloscope is a laboratory tool for analyzing real-time, recurring electrical signals, it is not adequate for servicing the requirements of microprocessors/microcomputers. The logic analyzer, a form of digital oscilloscope, has evolved as a convenient troubleshooting tool for microprocessor-based systems. It displays digital data in a convenient form, either as rectangular waveforms representing digital data or in a tabular binary form that can be analyzed by the flight-test engineer. Unlike analog real-time oscilloscopes or spectrum analyzers, the logic analyzer acquires and displays information in the digital domain. Information in the digital domain consists of binary data simultaneously presented in a bus (a group of parallel lines) as well as in the sequence these data change on these lines. The digital-displayed data also include the clock and control signals, which control the flow and processing of data and addresses.

In a conventional, non-storage oscilloscope, the instrument captures and displays events that occur after a trigger is applied. For stable display, the event(s) must be repetitive. In a logic analyzer, the events preceding the application of a trigger are captured, displayed, and need not be repetitive. With a conventional oscilloscope, the trigger initiates the capture and display of information. In a logic analyzer, the trigger can be used to capture and display information that is present just before a trigger is applied. This feature uses semiconductor memory devices for temporary data storage.

Most logic analyzers have the following characteristics:

1. They have several parallel input channels. Sixteen channels (or more) are quite common.
2. The incoming digital information is stored in semiconductor memory chips.
3. The incoming binary bits are stored on a single pass. Unlike the oscilloscope, the incoming information does not have to be recurring.
4. Logic analyzers capture and display information that arrived prior to the application of the trigger.
5. The incoming binary bits in all the input channels can be captured, stored, and displayed as selected by the operator.

The incoming signal pulses are fed into a threshold detector that detects and converts the signals to the proper binary levels required by the analyzer. In many instruments, the threshold voltage is selected by the operator to match the threshold voltages applicable to the logic family used in the device being tested. Most instruments have one fixed-threshold setting of 1.4 V to accommodate the standard TTL family of logic circuits. Additionally, the range selection varies from ± 2.5 to ± 12.0 V. Other fixed-threshold settings are also used. A threshold setting of -1.3 V is often used for testing high-speed emitter-coupled logic (ECL) circuits.

3.4.2 Emulators

During the system-development cycle, there is a time when the hardware and software must be joined. The sooner these are joined the better. With in-circuit emulation, software can be executed on the actual prototype hardware. This has two major advantages:

1. The software can be tested and debugged under realistic operating conditions.
2. The prototype hardware can be tested and debugged under the same conditions.

During this emulation task, the development system (or special in-circuit emulation device) imitates the microprocessor by executing the program in the native instruction set and machine code. Thus, the entire prototype hardware system (minus the actual microprocessor) can be tested and analyzed with various equipment, such as a logic analyzer.

The emulation task should be started after the prototype software has been written and translated, but before final system testing and documentation have been done. Final system testing and software validation should be done with the actual microprocessor in the system.

Possible sources of emulators and related equipment for a selected microprocessor include the microprocessor manufacturer; general instrument companies such as Hewlett-Packard, Fluke, Tektronix, Thomson-CSF; PROM programmers; and general-purpose computer manufacturers.

3.4.3 Microprocessor Development Systems

Since microprocessor-based systems are generally complex, different development systems have been devised so that programs may be developed with the aid of special-purpose software and peripherals. These programs can later be transferred to the actual systems on which they will run.

Microcomputer development systems consist of a microprocessor with additional hardware and software suited to the development tasks. The development systems can be used with a variety of microprocessors and have the following components:

1. Several work stations for multiple users
2. An editor for program changes
3. Assembler/compiler for program translation
4. Linker to integrate the program
5. A facility for changing and displaying the contents of memory locations
6. A reset control that starts the processor in a known state
7. A single-step control that allows a program to be executed one step at a time for debugging purposes
8. A run control that allows a program to be executed, beginning at a specified memory location
9. RAM that can be used as program memory
10. Interfaces to standard I/O devices, such as keyboards, LED displays, line or character printers, floppy-disk systems, and CRT displays.

These are other features provided with some development systems:

1. PASCAL, "C", FORTH, JOVIAL, ADA, FORTRAN, BASIC, PL/M compilers
2. Processor for specific emulation
3. A facility for setting breakpoints and initiating traces
4. Connectors for interfacing external devices to the processor buses
5. PROM/EPROM Programmer

The best source for microprocessor-based system development aids and equipment is the manufacturer of the selected microprocessor. Other sources include general-instrumentation companies. Many features of microprocessor development systems can be adapted to various microprocessors by using special plug-in boards. (The Hewlett-Packard HP 6400 Logic Analyzer and Development System is used with over 30 microprocessors from several manufacturers.)

3.5 SYSTEM ENVIRONMENT AND RELIABILITY

The reliability of a microprocessor-based system is a function of its storage and operating environment. Flight-test airborne-instrumentation systems are subject to special environments. Temperature extremes, as well as temperature shocks and cycling, mechanical shock and vibrating, electromagnetic interference (EMI), and altitude changes are all special environmental conditions that must be considered during the design, development, and testing of flight-test instrumentation (as well as avionics in general).

If a microprocessor or any device in the flight-test system is subject to environmental conditions beyond its specified limits, the reliability could be degraded. It is commonly accepted that relatively high vibration and/or temperature shock and cycling could cause integrated-circuit devices to fail because the internal wire bonds break or the wires sever. Relatively high or low temperatures can cause the semiconductor to change so that voltage and current operating ranges drift out of the specifications of the device.

Both operating and non-operating environments must be well understood if the specifications for the microprocessor chip set and other components are to be accurate. R.W. Borek, in an AGARDograph concerning instrumentation-system installation, outlined procedural principles that instrumentation-development engineers should consider before installing an electronic system onboard an aircraft. (Ref. 30). The environment's worst-case limits must be known prior to installation of flight-test instrumentation.

3.5.1 Environment Estimation

The best source for environment limit estimation at selected parts of the aircraft is the airframe manufacturer. Aircraft parts undergo different environmental conditions at different speed, altitude, and maneuvering ranges. If the environmental conditions can be determined, the correct flight-test instrumentation components can be chosen to keep the system in specification.

If the flight-instrumentation-design engineer cannot determine all the environmental conditions, general specifications for the aircraft and its avionics may be helpful for estimating environmental limits. Instrumentation installation must survive in the same environment as the system or subsystem it measures. Aircraft environmental specifications for airborne electronics equipment are published by the U.S. Department of Defense, MIL-E-5400T, (Ref. 31). This document discusses environments up to 100,000 feet (31,250 meters) and temperature limits of -54°C to +125°C. Also discussed are vibration limits for propeller and jet-powered aircraft, as well as helicopters. Other publications that may be used for environmental estimation are Environmental Test Methods, MIL-STD-810C (Ref. 32), Climatic Extremes for Military Equipment, MIL-STD-210B (Ref. 33), and Environmental Conditions and Test Procedures for Airborne Equipment (Ref. 34). These documents discuss a wide range of airborne environmental tests.

3.5.2 Reliability

Reliability is a consideration at all levels of electronics from basic materials to full operating systems. Reliability is an engineering discipline that needs attention early in the design phase, as well as throughout instrumentation-system development, installation, and final testing. Reliability should be fundamental in any flight-test program. Each hour of a flight test is critical in terms of resources and flight safety. Sometimes a system is considered reliable because it contains military-qualified components, or a reliability engineer has analyzed it after the fact. This thinking can lead to wasted flight hours and costly system redesign.

There are many factors in airborne-electronic-equipment reliability: system design, component selection, environmental conditions, testing, and handling. Two papers on reliability engineering, Coppola (Ref. 35) and O'Connor (Ref. 36), discuss these topics in detail. Reliability screening of integrated circuits began in the mid 1960s. (Ref. 35). Several testing standards and methods have been developed since then. Test Methods and Procedures for Microelectronics, MIL-STD-883C (Ref. 37) and General Specifications for Microcircuits, MIL-M-38510F (Ref. 38) define uniform methods and procedures for testing integrated circuits. Also defined are processing procedures in chip fabrication. Microprocessors and support integrated circuits are fabricated and tested based on these two specifications. A good document to aid in reliability calculations is a reliability prediction handbook. (Ref. 39).

The decision on the use of MIL-SPEC devices must be made early in the flight-system design process. A primary driver in the decision is the application environment. Since MIL-SPEC devices are quite expensive, the decision can greatly affect development costs. If it is decided to use MIL-SPEC devices in the final airborne system, commercial SPEC devices could possibly be used in the development/prototyping stages to lower costs. Electrical specifications must be examined very closely since MIL-SPEC and their commercial part substitutions are not electrically identical.

Most integrated circuits are not fabricated to military specifications; some are not fully tested before they are shipped from the manufacturer. A very small number of microprocessors and support chips cannot function properly even before they are used in normal operation. Thus, before final system-design decisions are made, the flight-test instrumentation engineer must carefully consider the level of component quality needed to be consistent with the airborne environment.

Acceptable quality level (AQL) is a measure of the average level of defective devices within a sample. The average device quality will be a function of the fabrication and test-screening processes. An AQL of 0.05-percent to 0.1-percent defective devices within a sample is now reasonable to expect (but not to assume). This can directly relate to choice of semiconductor devices for the flight-test system.

3.5.3 Failure Mechanisms

There can be several reasons why an integrated circuit fails: destructive environments, inferior materials, inadequate process controls, and improper handling. A costly hazard, when handling microprocessors and MOS integrated circuits, is electrostatic discharge (ESD). Due to fabrication methods, most MOS microprocessors and related support devices are susceptible to ESD. Improper use of devices allows ESD to destroy millions of dollars' worth of devices every year. (Ref. 35).

The reliability of integrated circuits conforms to the "bathtub" curve, which plots failure rate as a function of time. As shown in Figure 10, the three phases associated with this curve are early life, useful life, and wearout.

Early Life (Infant Mortality): This phase covers devices that fail in the early life of the system. The failure rate decreases rapidly during this initial period, and the phase is short (a few weeks or months). The infant mortality varies widely from one application to another and is influenced by system-stress conditions.

Useful Life: This is the period extending from the end of infant mortality to the beginning of wearout, at which point the failure rate begins to increase again. The useful life is usually expressed in years. Although there is insufficient data to define a true useful life, in most cases, the minimum life of a device usually suffices to assure adequate design margins. The useful life usually extends for decades if adequate design margins are applied. Temperature plays a major role in triggering the beginning of wearout, but other stresses, such as pressure, mechanical stress, thermal cycling, and electrical loads are important. Failure rate is the percentage of devices that fail per unit time during the flat portion of the curve (from the end of infant mortality to wearout). Usually expressed as percent per 1,000 hours (sometimes per billion hours or FITs) this statistic can also be stated as mean time between failures (MTBF), which is simply the reciprocal of failure rate.

Wearout: The end of the useful life for a group of integrated circuits is characterized by an increasing failure rate as a function of time. Most integrated circuit devices, in the proper environment, will have several years of normal operation. While an accurate prediction is difficult, Intel has stated a goal of 20 years' useful life for their MOS dynamic RAM devices. (Ref. 40).

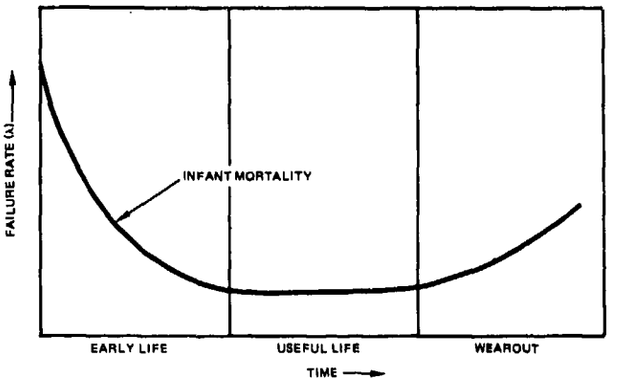


Figure 10. Integrated-circuit reliability life.

Inadequate fabrication processing and poor materials may cause over 50 types of failures to the integrated-circuit chip. (Ref. 41). These are some examples of principal failure mechanisms:

1. Encapsulation failure (humidity and imparity penetration)
2. Macrochip failure (cracks across the chip, small cracks at the bondings)
3. Imperfect chip attachment to substrate
4. Wire bonds (excessive bonding pressure, misplaced bonds, crossed wires)
5. Aluminum and gold conductors (electromigration, corrosion)
6. Design defects (diffusion faults, improper design)

Table 2 shows that the main stress factors contributing to infant mortality are electrical transients, noise, mechanical maltreatment, and excessive temperatures. Most infant failures occur in three stress-producing phases:

1. Device burn-in (a pre-operational screening process)
2. Card assembly and handling
3. Initial system test and operation

Table 2. Common infant mortality failure mechanisms.

Failure Mechanism	Defect	Stress Factors
Oxide ruptures	Thin or defective oxide (masking and oxidation)	• System electrical noise • System power interruptions • Inductive loading
Open-wire bonds	Assembly defects	• Ultrasonic exposure during card assembly • Excessive temperature
Lifted-die bonds	Assembly defects	• Excessive temperature
Fused-die Metallization		
Shorts	Inadequate spacing between adjacent stripes	• System electrical noise • System power interruptions
Open	Inadequate stripe-width and/or thickness	• Inductive loading
Corrosion of wire bonds and/or die metallization	Seal leaks (defective encapsulators)	• Handling damage • Excessive solder heat during card assembly

Although device burn-in can serve as an indicator of infant mortality, carefully controlled burn-in tests seldom resemble the broad range of conditions (especially electrical noise) encountered in actual use. Field reliability data is therefore taken as the best source of infant mortality.

3.5.4 Temperature Effects

A primary factor of integrated-circuit reliability is the junction temperature (not environmental temperature) on the device chip. The difference between the junction and environmental temperatures is a function of the total thermal resistance between the junctions and the ambient temperature.

The reliability will decrease exponentially with temperature increases in the chip junction. The following formula (Refs. 39, 42) is used to characterize the effect on an integrated circuit.

$$\frac{\lambda_2}{\lambda_1} = F = e^{-E/K\left(1/T_2 - 1/T_1\right)}$$

where

λ_1 = Failure rate at junction temperature T_1

λ_2 = Failure rate at junction temperature T_2

F = Acceleration factor of failure rate

T_1 and T_2 = junction temperatures in °K

E = Thermal-activation energy in electron units (eV)

K = Boltzman's constant (8.617×10^{-5} eV/°K).

The thermal-activation energy level is a function of chip-fabrication factors, but it can generally be estimated at 0.7 eV for NMOS and CMOS transistors and at 0.4 eV for bipolar devices such as the common TTL family of logic.

Equivalent junction temperature can be estimated by applying both the thermal resistance from the junction to the ambient air and the total power dissipated by the device to the following formula:

$$T_J = P_D \quad \theta_{JA} + T_A$$

where

T_J = Equivalent chip-junction temperature

P_D = Total chip power dissipated

θ_{JA} = Thermal resistance from junction to ambient air (normally determined by the device manufacturer)

T_A = Temperature of the ambient air

The principle of increasing temperature in order to increase the failure rate is used by the manufacturer to accelerate testing of the device. Microprocessors are tested at a high temperature (125°C) to stress the devices. Failure rates can be estimated from the test results. Recent tests by Motorola indicate their CMOS microprocessors have a failure rate of 1.3×10^{-7} (failures/hr) or (0.013%/1,000 hr) at an equivalent operating temperature of 70°C. (Ref. 43).

4. SELECTED APPLICATIONS OF AIRBORNE MICROPROCESSORS IN FLIGHT-TEST PROGRAMS

Microprocessor applications in flight-test equipment are numerous, and a comprehensive application picture would be very difficult to describe. Microprocessors are currently playing expanding roles in these flight test areas: data logging, signal processing, displays, system diagnostics, controls, and calibration.

This section describes some selected applications of interest to engineers working with microprocessors in airborne systems.

The following descriptions represent examples of how microprocessors are used in flight testing and flight-test instrumentation. The author wishes to thank the instrumentation engineers whose valuable contributions provided the primary source material for this section. Their names appear at the end of each application description.

4.1 AIRBORNE DATA SYSTEM WITH ONBOARD REAL-TIME ANALYSIS (LOCKHEED GEORGIA)

Lockheed Georgia Company, Marietta, Georgia, has developed a flight-test data system that collects, processes, and displays real-time data using several distributed microprocessors. The system was developed for use on the C-141B and has been used on other transport-type aircraft. The Lockheed system enables the flight-test engineer to look at various element groups of processed data during the flight.

This real-time data can be rapidly evaluated during selected tests as the flight progresses. This reduces flight time and costs since the acceptability of a maneuver can be immediately determined from the onboard data. Also, the possibility of later reflights due to missed data is reduced.

The basic data-collection system consists of an acquisition microcomputer and one to eight remote input modules. Each of these input modules can signal-condition 16 sensor channels. The microprocessor-based acquisition computer has a 128-channel capability. If required, up to eight acquisition computers may be connected to provide 1,024 data-measurement direct channels.

The Texas Instruments TI9900 microprocessor(s) allows the acquisition computer to control the input modules acquiring the data. This microprocessor performs offset corrections and multiplications to convert sensor scale factors to engineering units. It also provides a form of dynamic sensor calibration. The microprocessor accomplishes several parameter conversions such as time intervals into rates, period measurements into frequencies, discrete strain measurements into loads, and voltage into temperatures. Acquisition computers perform distributed processing functions throughout the aircraft. They are operated under the control of a master computer, which also uses a TI9900 microprocessor. The master computer allows the acquisition units to independently perform both distributed data acquisition and parameter-conversion processing. The master computer also controls the transfer of this preprocessed data to the main memory. The transfer occurs over several hundred times a second.

A complete scan of 1,024 data channels requires 372 microseconds. Figure 11 shows a functional diagram of the Lockheed Airborne Data System (LADS). The main function of the LADS system, in addition to logging flight-test data, is to process the data under the direction of an applications minicomputer. (Ref. 9). The applications computer (a coprocessor with the master computer) allows the flight-test engineer to select an application of current interest. Application programs are used to process the flight-test data and present the results as a graphical, real-time display. Programs are executed in the applications computer, using processed data from the master computer. A good example of an applications program is the one in which aircraft gross weight and center of gravity are displayed in real-time. The program reads 10 fuel-tank quantities, applies corrections, computes fuel available, computes current gross weight and total moment (considering flap and gear configuration), then computes current center of gravity and stall speed. The flight-test engineer can see this and other important information displayed in succession as the aircraft is maneuvering.

The LADS system is cost-effective because it utilizes microprocessor-based, distributed data processing. A great deal of digital processing must be accomplished as physically close to the distributed sensors as possible. Without microprocessors, this concept would be extremely costly.

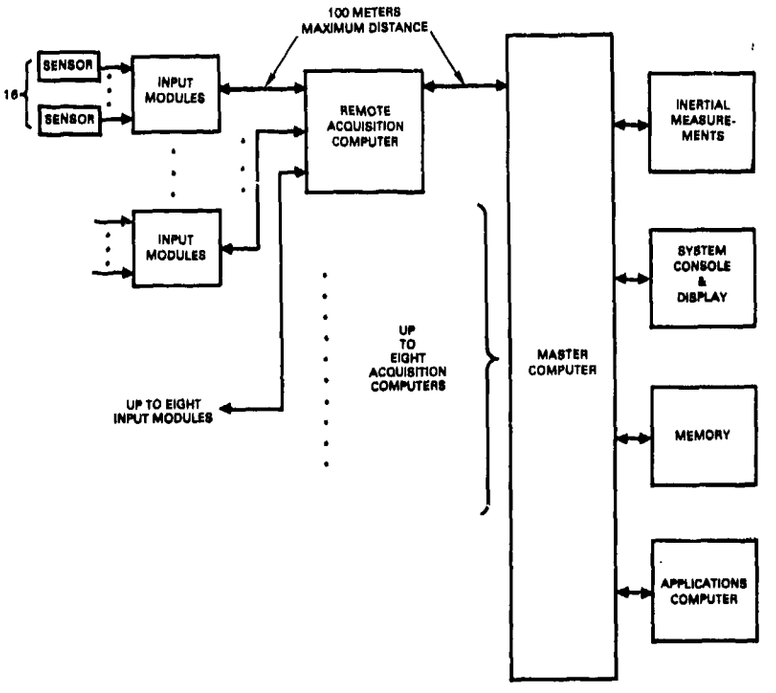


Figure 11. Lockheed Airborne Data System (LADS).

4.1.1 Microprocessor-Related Functions

The acquisition microcomputer uses a Texas Instruments TMS9900 16-bit microprocessor with static RAM, EPROM, and a DMA control device. The digitized sensor data is stored in RAM under DMA control. DMA control is used to input data and commands to the read/write memory with the least time delay. The TMS9900 microprocessor is used as the controlling and computing element for each group of 16-channel modules. The acquisition microcomputer has 4-K (4,096) words of read/write memory (RAM) and 4-K words of program-stored, non-volatile memory (EPROM).

The software is written in assembly language to optimize speed and memory space. All arithmetic operations are done in fixed-point format to achieve the necessary processing speed.

There are three levels of interrupt control. Interrupt level zero is an unmaskable, power-up interrupt. When power is applied to the system, it vectors program execution to the initialization routines of the TMS9900 internal registers and interval timer. Another interrupt level synchronizes the eight acquisition computers to the master-computer commands for the sensor scans. The third interrupt level signals the acquisition processor that DMA transfers have been received from the master computer for configuration changes.

Input data are sampled under CPU control at rates ranging from 20 to 160 samples per second. These data are acquired by signal-conditioning modules during a sampling window 375 microseconds wide. The data are transferred into RAM by DMA. Once the CPU senses data-transfer completion, it begins to process raw data into scaled engineering units. All direct data are processed first. Derived measurement calculations then follow, but only for those derived measurements specified in the database. All processing is completed prior to initiation of the next data sample; idle time is used in servicing the CRT display.

The Lockheed flight-test system is being updated to use the Intel family of microprocessor-related devices, including the 16-bit 8086 microprocessor and the 8087 math coprocessor for increased computing speed. Selected sections of the system memory will consist of EEPROM devices (due to its in-circuit programming and nonvolatile features) and greater use of Large Scale Integration (LSI) for support devices. The software will be developed with PL/M. Selected program portions will be written in assembly language.

Technical Sources: Mr. James A. Tabb, Staff Engineering Specialist and Mr. Michael L. Roginsky, Engineering Specialist, Engineering Test and Evaluation Division, Lockheed-Georgia Co., Marietta, Georgia.

4.2 AIRFRAME FLUTTER TESTING (MCDONNELL AIRCRAFT)

McDonnell Aircraft Company of Saint Louis, Missouri (a subsidiary of the McDonnell Douglas Corporation) has developed a microprocessor-based system to assist in determining airframe flutter characteristics of fighter-sized jet aircraft. The system is called the Flutter Exciter Control Unit (FECU). This flutter exciter control system, with some modifications, is used in the flight testing of several different aircraft. It interfaces with flight-control computers to provide both analog and discrete command signals, resulting in controlled deflections of various aircraft surfaces. These deflections of the ailerons, the stabilators, and the rudder induce a controlled, airframe flutter. Flutter characteristics can then be studied and evaluated. Flight safety is a paramount consideration with this type of testing.

The flutter exciter control system is a programmable, digitally controlled signal generator. The system is developed around a CMOS 8-bit microprocessor, the RCA 1802.

The RCA microprocessor performs overall control and monitoring functions based on an operating system (firmware) stored in EPROM. Waveform generation and signal generation are assigned to peripheral devices rather than the microprocessor. The waveform characteristics are also stored in EPROMs and are selected by the microprocessor, based on prestored user inputs. The design is inherently flexible and can provide various waveforms to test different airframes, or it can adapt waveforms when modifications of a given airframe are necessary.

The control-unit panel keyboard is located in the cockpit so the test pilot can control and monitor the FECU. Figure 12 shows the FECU control panel. Amplitude, frequency, sweep rate, mode, and aircraft aerodynamic control surfaces can be controlled and monitored by the pilot as events are executed by the microprocessor. The signal-output circuitry can scale three waveforms for the selected control surface. To ensure flight safety, shutdown-detection logic provides monitoring of several internal and external fault conditions.

4.2.1 Flutter Excitation (Flight-Testing Background)

Testing for flight flutter involves exciting the aircraft's natural vibration modes during selected flight conditions and measuring the response to the excitation. The critical flutter speed is then predicted from measurements acquired at subcritical speeds. (Ref. 44).

In the early years of flight testing, the primary in-flight flutter excitation technique was pilot-induced stick raps and rudder kicks. By the mid-1960s, McDonnell and other aircraft companies were using electronics to provide the necessary forcing functions on the aircraft control surfaces to cause flutter. As microprocessor-based systems evolve, so will the flexibility and precision of how well the flutter modes can be generated, controlled, and measured.

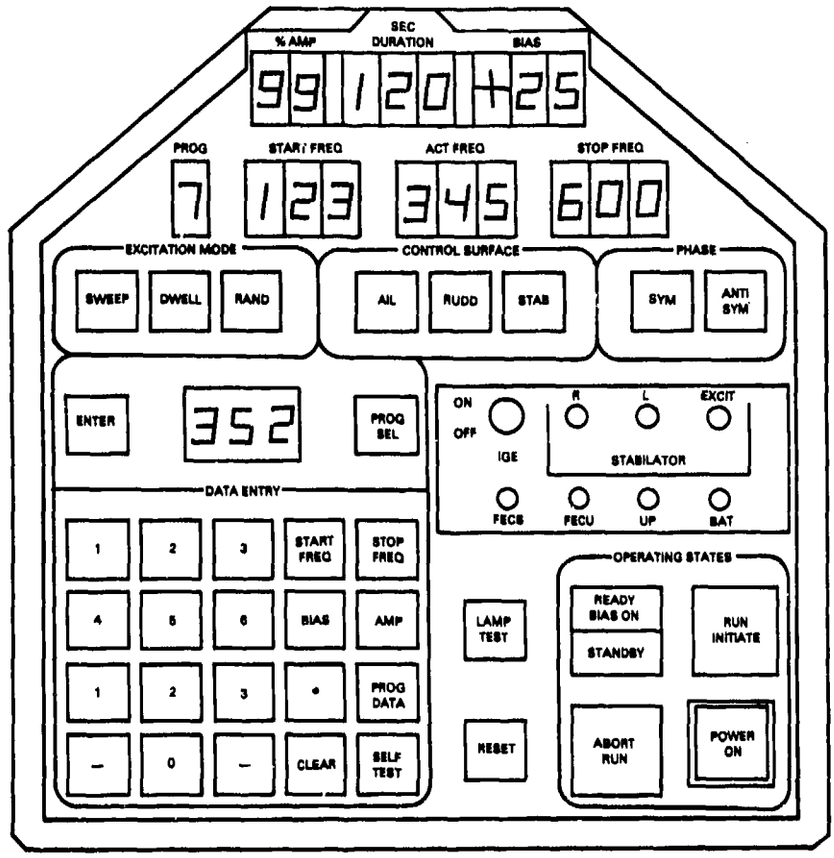


Figure 12. Front panel, FECU.

A basic factor in the design of new aircraft is freedom from flutter throughout the entire flight envelope. With higher speeds, lighter materials, and diverse wing configurations, flutter considerations are even more critical. In spite of advances in theoretical flutter analysis and wind tunnel measurements, actual flight tests are still necessary to verify that an aircraft is free from flutter within a given flight envelope.

When a wing or other structure in an airstream is subjected to a disturbance, the resulting oscillatory motion induces changing aerodynamic forces on the structure. At low speeds, these forces tend to oppose the motion, thus providing positive damping of the structure. However, as the speed of the aircraft increases, the forces on the structure change. This can cause new vibration modes. (Ref. 44, 45).

To accurately characterize the flutter behavior of the aircraft, good quantitative frequency and damping data must be obtained for all modes of interest. Since flight testing is very costly, flutter-testing programs must be structured to be comprehensive, accurate, and time-efficient; yet, the aerodynamics and structural characteristics of the aircraft cannot be changed. (Refs. 46, 47). A microprocessor-based system such as the FECU can achieve these objectives.

Figure 13 illustrates the basic functions of flutter excitation. Table 3 compares two exciter systems: the first is a mid-1960s electronic system, and the second is a newer FECU. Both were used for fighter-sized jet aircraft.

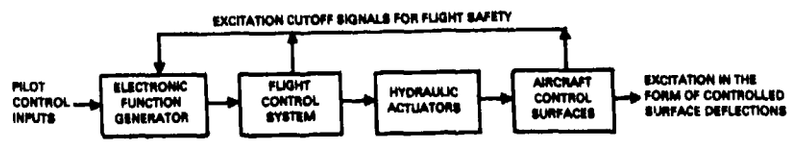


Figure 13. Flutter excitation using an electronic function generator.

Table 3. Characteristics of electronic flutter exciters.

	Stabilator exciter only	FECU
Control-surfaces excited	Stabilator	Ailerons Rudders Stabilators
Electronic-device technology	Discrete components (transistors, etc.)	CMOS (digital and analog)
Excitation frequency range and accuracy	8 Hz to 30 Hz ± 0.33 Hz	2 Hz to 60 Hz ± 0.01 Hz
Full-scale surface deflection	± 2°	Ail: ± 10° Rud: ± 10° Stab: ± 10°
Forms of excitation	Square wave	Sinusoidal Random
Sweep-rate control	Internally set to rate	EPROM programmable
Control of excitation	Manual	Stored programs
Visual cues for pilot	7	28
Adjustable/Programmable parameters	9	22
No. of stored programs/ Method of program entry	N/A	16 programs entered via front-panel keyboard, before or during flight
In-flight excitation setup (typical time, seconds)	Potentiometer and switch adjustments (30 s)	Auto-increment (1 s) Program recall (5 s) New program entry (30 s)

4.2.2 Characteristics of the Flutter Exciter

The FECU was designed primarily using CMOS integrated circuits, both digital and analog. The unit supplies excitation as well as bias to the ailerons, stabilators, and rudders via a flight-control computer. The excitation can be swept sinusoid or constant sinusoid, or it can be a bandpass-limited, random signal. Excitation to the ailerons and stabilators can be symmetrical or antisymmetrical.

Excitation parameters are selected in complete sets called "programs." Each program contains all the characteristics required to specify an output excitation. Programmable parameters include surface (aileron, rudder, stabilator); mode (sweep, dwell, or random); start and stop frequency (2.0 to 60.0 Hz); percent full-scale amplitude; symmetry; bias deflection; and run duration. Programs are entered, stored, and recalled from a front-panel keyboard. The selected program parameters appear in light-emitting diode (LED) displays on the front panel. Up to 16 such programs can be stored in a nonvolatile memory within the FECU. In the sweep mode, the duration is computed (as a function of start and stop frequencies) and displayed. The FECU has three operating states: standby, ready, and run. In the standby state, no excitation is output. In the ready state, the programmed bias is applied at 1 degree per second and held. In the run state, the programmed excitation is applied. Runs can be halted at any time by pressing the ABORT RUN switch. The FECU also can be disengaged by a paddle switch on the control stick. Safety-of-flight inputs to the FECU include shutdown signals from the flight-control computer or strain-gage outputs caused by excessive loading of the stabilators.

The FECU performs a self-test function. Storage of excitation parameters in the form of programs reduces pilot workload by eliminating the need to set front-panel controls. This allows more testing per flight. The programs anticipated for use on a test flight can be entered as part of the preflight procedure. Programs are then executed sequentially during the desired flight conditions.

4.2.3 The Microprocessor

The RCA 1802 microprocessor, used as the FECU's CPU, is fabricated with CMOS technology to keep power-supply requirements to a minimum. CMOS devices also have the advantage of substantial noise immunity. Most instructions are single byte and are executed in 3.2 microseconds with a 5 MHz clock. If the 1802 is operated with a clock of 5 MHz and operated at 5 Vdc, it requires only 4 mA from the power supply (20 milliwatts). This RCA microprocessor is fully static and can be operated with any clock frequency up to approximately 6 MHz. In the FECU design, a lower clock frequency was used in order to simplify and reduce power in the memory and I/O circuitry. The 1802 microprocessor can be operated over a range of voltages (4 to 10 Vdc) and over the normal military temperature range of -55°C to + 125°C.

The 1802 has a bi-directional 8-bit data bus, 8-bit multiplexed address bus, programmed I/O, DMA, and one-level vectored interrupt. This architecture was modified in the FECU by additional hardware to provide a 16-bit buffered data bus, buffered address bus, isolated memory and input busses, modified-memory mapped I/O, and expanded timing and control signals. Figure 14 is a block diagram of the FECU system.

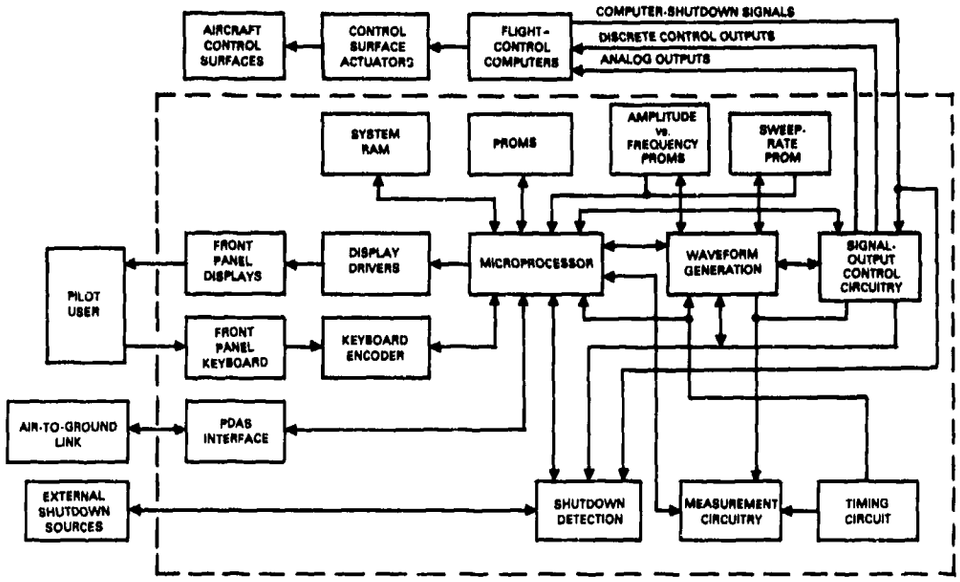


Figure 14. FECU system.

4.2.4 Firmware

The operation of the Flutter Exciter is controlled by the microprocessor as it executes PROM stored firmware programs.

The firmware programs perform these main functions:

1. Process and store keyboard inputs into valid parameter sets (programs) in order to define a specific mode of operation.
2. Recall and verify previously stored programs.
3. Update panel displays and air-to-ground interface.
4. Control hardware functions through the generation of control words and/or control strobes.
5. Manage operating states through monitoring of internal parameters, external inputs, and manual key entries.
6. Detect faults on a limited, but continuous basis, with a more extensive, dedicated self-test function available on command.

The firmware controlling the dedicated self-test function is not contained with the main firmware but is stored in the sweep-rate PROM. This allows changes to the self-test limits without the need to alter the main firmware. This firmware is down-loaded into RAM for execution. After completion of the self-test cycle, the set self-test routine is purged from memory. A failure during self-test generates an interrupt that forces the executions back to the main firmware, which utilizes existing displays to indicate failing functions.

Technical Sources: Mr. Robert L. Grant, Technical Specialist and Mr. James J. Meany, Senior Engineer, Fight Test Data Acquisition, McDonnell Aircraft, Saint Louis, Missouri.

4.3 IN-FLIGHT FAILURE SIMULATION SYSTEM (AAEE, UK)

The Aeroplane and Armament Experimental Establishment, Boscombe Down, United Kingdom, has developed an in-flight Failure Simulation System (FSS) for use in a special Tornado aeroplane. (Refs. 48, 49, 50). The FSS is designed to inject test signals into the aircraft flight-control system to test its response to simulated hardware failures. The system is used to test the combined behavior of the pilot as well as the aircraft during the recovery period.

The FSS uses both analog and digital signals to simulate a failure. After the simulated failure has had its effect, the system measures and displays the time necessary for the pilot to recognize the simulated failure. The full scale of this time display is 99.9 seconds with 0.1-second resolution. This measured, elapsed time will be displayed until the pilot initiates a new action. The pilot can disconnect the FSS from the flight-control system simply by pressing an abort switch or switching FSS off. The FSS does not affect the flight of the aircraft, but it directly manipulates autopilot signals and is therefore potentially a safety critical system. A primary design goal was an adequate level of integrity with a reasonable cost.

Flight-safety integrity was achieved by having all critical circuits of the FSS duplicated and isolated, so as to prevent self-failure from inducing a safety critical condition. Figure 15 is a function diagram of the FSS. Neither the analog nor digital channel of the duplex system can output signals to the Autopilot and Flight Director System (AFDS) without authorization by the other channel. Any detected failure arising within critical areas of the FSS circuitry causes a rapid disconnect of the FSS electronics from AFDS and restores all AFDS signal connections to normal.

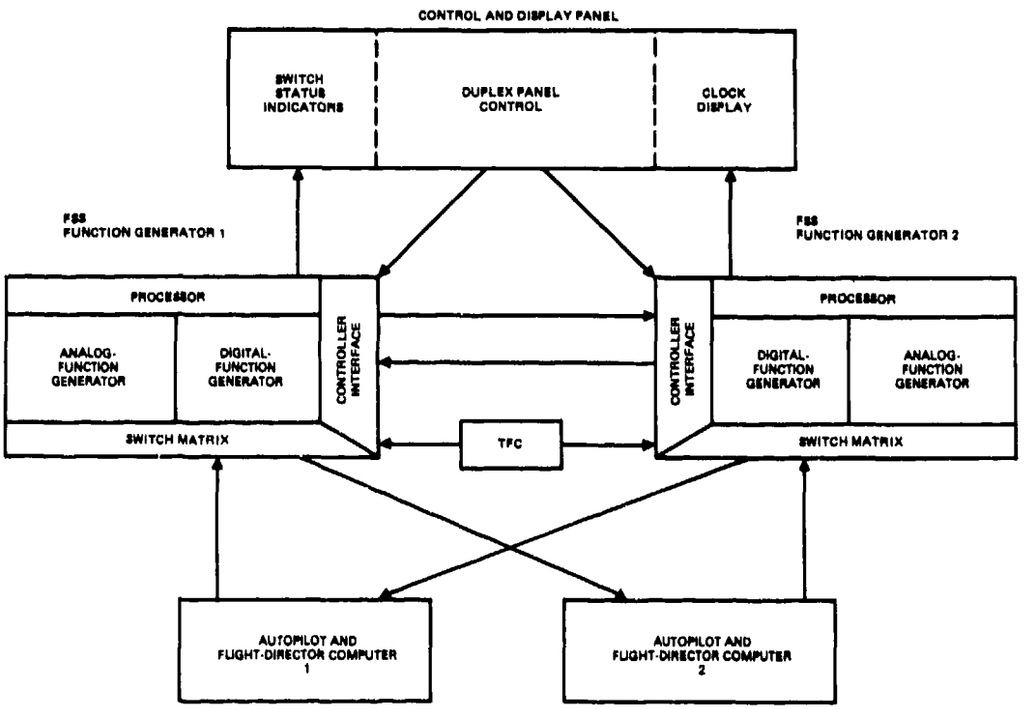


Figure 15. Failure Simulation System.

The FSS consists of three units. The Failure System Controller (FSC), includes the display panel used by the rear cockpit operator. The FSC is a slave peripheral to two other units, called Failure Simulation Function Generators (FSFGs). These two units are identical and interchangeable. They are characterized by external wiring into a master (FSFG1) and slave (FSFG2) relationship. This characterization causes a slightly different software configuration to be adopted by the master and slave. The FSFGs are responsible for virtually all FSS operations. They interpret, echo operator commands, generate the necessary AFDS signals, and connect these as appropriate to the AFDS. A major part of the AFDS is its two computers. The functional connections of these two computers and the Terrain Following Computer (TFC) to the FSS are shown in Figure 15. To minimize ground currents that would introduce errors in the analog cross-feed signals, each FSFG is powered via an isolating dc-dc converter, and all signal lines between the FSFG and other FSS units preserve ground isolation (in most cases by the use of optical coupling).

Each FSFG contains five functional modules:

1. A microprocessor-based controller
2. An interface to the FSC
3. An analog-function generator
4. A digital-function generator
5. Relay switching circuits that connect the FSS to AFDS wiring

The processor module is based on Intel's 8085 microprocessor. The program and fixed data are stored in 16K bytes of EPROM. All unused EPROM locations are filled with HALT instructions to improve integrity. Read/write memory receives 1K of static RAM. The module also provides other hardware utilities such as timer/counter functions and interrupt facilities. (See Figure 16.) The I/O structure of the processor is connected to all other modules via a common backplane.

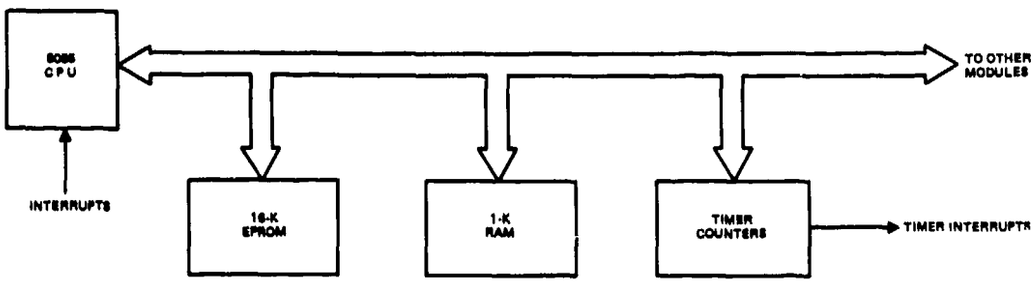


Figure 16. Processor module.

4.3.1 Flight Safety Precautions

Since FSS operation is potentially critical to the flight safety of the aircraft, the self-test and system-abort features should be summarized. A processor reset occurs at power-up, whenever the 5-V supply falls below its minimum threshold of 4.5V, or whenever the TEST switch is operated. The reset initiates all hardware, and its removal causes an orderly start of program execution from memory location zero. A power-up or power-fail reset always latches an ABORT condition, isolating the FSS from the AFDS. To resume operation, the operator must press TEST. After resolving its master/slave role, the processor enters a self-test mode and checks these items:

1. RAM
2. EPROM
3. Power supplies
4. Relay drivers
5. Analog and digital-function generators
6. Watchdog and other timers
7. Abort circuits

Upon successful completion of the self-test mode, the panel display is initiated and controls are scanned for operator-command inputs. Failure of the self-test causes an ABORT condition; the AFDS is disconnected from the FSS circuits; and an error code is displayed indicating the location or type of failure.

The FSS abort system disconnects the AFDS from the FSS in the event of internal failure or an operator ABORT request. ABORT action is initiated by any of the following:

1. Operator pressing ABORT
2. A watchdog time-out
3. 5-V power supply failure
4. A CPU-output instruction in response to a software-detected failure of hardware
5. An ABORT by the other FSFG

With any such event, the ABORT latch activates the ABORT condition until the TEST switch is operated. The ABORT latch is not defeated by loss of power. Each FSFG has its own ABORT circuit. Circuits are coupled so that an ABORT of one causes the other to ABORT, and a failure of one ABORT circuit cannot prevent an ABORT by the other.

An independent watchdog timer detects CPU runaway conditions. Timing independent of the processor clock can check the duration of the main program cycle. Maximum and minimum limits are imposed and any limit exceedance causes an ABORT. The CPU can test and set a subsidiary latch to determine whether the watchdog has tripped, but the CPU cannot reset the ABORT condition unless the TEST switch is depressed.

Handshaking lines are provided between the two FSFGs to synchronize their operation. To improve integrity, time-out constraints are placed on the handshake sequence by each processor.

The FSC interface module provides input conditioning for Stick Force Cut-Out (SFCO) and Instinctive Cut-Off (ICO) signals. These signals are buffered and protected against computer failure. Three signals are recognized: the ICO, SFCO (Pitch), and SFCO (Roll). All are handled in the same way. They initiate termination of an existing fault-generation sequence and stop the display clock from incrementing.

4.3.2 The Microprocessor

The Intel 8085 microprocessor, used as the CPU of the FSS, is an improved version of the popular 8080 microprocessor. Both the 8085 and 8080 are 8-bit NMOS microprocessors. Intel specifies that the 8085 is 100-percent software compatible with the 8080, but it is not hardware pin-for-pin compatible.

The 8085 has an on-chip clock and an instruction-cycle time of approximately 1.5 microseconds. The microprocessor also has four vectored-interrupt inputs, serial I/O features, and is powered with a single 5-Vdc power supply.

The 8085 controls the FSS by means of a stored program within the 16K EPROM. System status signals and data, including those from the operator, are inputted to the 8085 data bus, processed, and transferred to appropriate units by the microprocessor.

Technical Source: Mr. Allen Wright, Head of CIG Instrumentation Design, Aeroplane and Armament Experimental Establishment, Boscombe Down, United Kingdom.

4.4 ARINC DISPLAY SYSTEM (NLR, NETHERLANDS)

The National Aerospace Laboratory (NLR, Amsterdam, Netherlands) has developed an ARINC Display System that can display two parameters from an ARINC 429 data stream simultaneously. (Ref. 51). The system is designed as a development tool for the testing and maintenance of flight test instrumentation. It is designed to flight-test instrumentation standards, for use during flight. Although several such display units are on the market, they do not incorporate a microprocessor. Nevertheless, the specification for this instrument requires the following special features that can only be realized by a microprocessor:

1. The displays must present the labels, the data words, and the designation matrix of one or two parameters selected from one ARINC 429 stream.
2. The data must be presented either in engineering units (calibrated data) or in hexadecimal form (rough data).
3. The labels must be presented either in mnemonics or in octal code.
4. Parameter selection must take place via a keyboard, either in octal code (numeric presentation) or by function keys (mnemonic presentation).
5. It must be possible to change the calibrations for the presentation of data in engineering units.

Figure 17 shows the functional operation of the ARINC Display System. Both the ARINC labels to be presented and the method of presentation are chosen using the 16-key keyboard. The ARINC interface enters successive ARINC words into the microprocessor. Only ARINC words with the selected labels are further processed. The processed data are then transmitted to the correct display by the display interface driver.

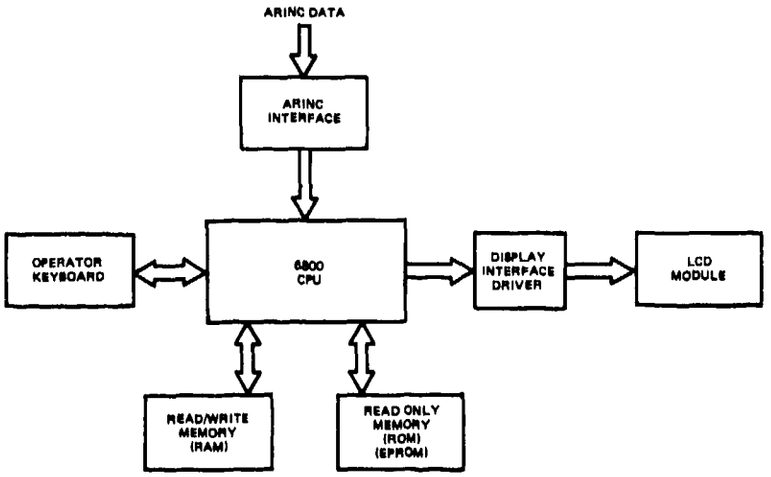


Figure 17. NLR-ARINC Display System.

The ARINC Display System is based on the Motorola 6800 microprocessor (with an extended temperature range for use onboard aircraft). The 6800 has an 8-bit data bus, a 16-bit address that can directly address 65,536 addresses, and vectored interrupts. It is powered by a single 5-V power supply. It has 72 variable-length instructions with seven addressing modes. The instruction set resembles that of the PDP-11 minicomputer.

The program is stored in a 2K-byte EPROM; temporary data (calibrations and keyboard inputs) are in a 1K-byte RAM. The keyboard and the liquid-crystal display are connected to the microprocessor via a standard Peripheral Interface Adaptor (PIA). When the system was designed, no standard interfaces for ARINC busses were on the market. This interface was, therefore, specially designed by the NLR. (See Figure 18.) Each entering bipolar ARINC word is transformed into digital TTL-level code and then written into the 32-bit shift register. When the shift register is full, a sync pulse is sent to the interrupt

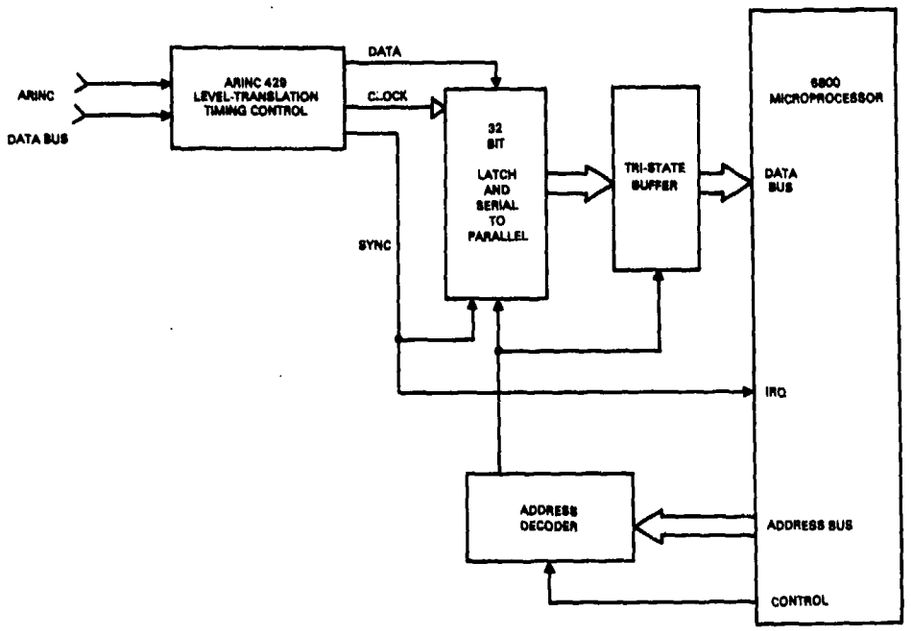


Figure 18. ARINC data bus and microprocessor interface.

Request (IRQ) input of the microprocessor. Four address codes are provided on the address bus, and the ARINC code is read as four bytes. If the label of the ARINC word matches one of the selected labels, the word is processed to provide the selected mode of presentation.

The LCDS used in the system (Hitatchi H2570 series) display alphanumeric characters in a dot-matrix format. This type of display offers several important advantages: Power consumption is exceptionally low (0.5 mA @ 5 Vdc); long life is inherent, since the unit is solid state; there are 200 different character fonts; weight is approximately about 25 grams; and the contrast ratio of the display remains the same under all ambient light conditions.

Technical Sources: Peter J. Manders, P. G. Beerthuizen, and Jan M. Visser of the Information Division, National Aerospace Laboratory, The Netherlands.

4.5 STALL-SPEED WARNING SYSTEM (NASA)

The NASA Dryden Flight Research Facility at Edwards Air Force Base, California, has developed an aircraft Stall-Speed Warning System that is based on the Intel 8085A microprocessor. NASA designed, built, and flight-tested this system jointly with the U.S. Army for the Army's OV-1C Mohawk aircraft, but the approach that was used could be applied to several other aircraft. (Refs. 52, 53, 54). The system displays both airspeed and stall speed to the pilot, and it presents a synthesized voice to warn the pilot at a predetermined safety-margin speed.

The microprocessor combines several aerodynamic parameters to compute the airspeed and stall speed in real time. These parameters include dynamic pressure, altitude pressure, elevator and flap positions, engine torques, horizontal and vertical acceleration, and fuel used. Eleven sensors are used to measure these flight parameters. (See Table 4.)

With the support of an arithmetic processor (coprocessor) unit (Intel 8231A), the Intel 8085A 8-bit microprocessor can perform computations fast enough to present airspeed and stall speed to the pilot in real time. The special purpose arithmetic processor performs intensive operations such as trigonometric and inverse-trigonometric functions, square roots, logarithms, and the more typical functions such as addition, subtraction, multiplication, and division. These arithmetic operations can be done in either of two formats: fixed point, single and double precision (16 or 32 bits at a time) or in 32-bit floating point. The Intel 8231A uses special-purpose instructions that execute these arithmetic functions and formats with a speed advantage several hundred times faster than the general-purpose 8085A, which acts as the data controller to the 8231A. The two devices exchange data via their common data bus.

4.5.1 Speed Calculations

Equation 1 gives indicated airspeed (VIFPS), feet per second:

$$VIFPS = \left[\frac{2(70.7262)\ lb/ft^2/inHg\ Q}{p_0}\right]^{1/2} \qquad (1)$$

Table 4. Sensors and flight parameters for the Stall-Speed Warning System.

Sensor	Parameters	Units
S0	Longitudinal acceleration	g
S1	Vertical acceleration	g
S2	Flap position	deg
S3	Elevator position	deg
S4	Landing-gear, strut-compressed switch	
S5	Pressure altitude	inHg
S6	Dynamic pressure	inHg
S7	Total fuel used, right engine	gal
S8	Total fuel used, left engine	gal
S9	Left-engine torque sensor	lb/in^2
S10	Right-engine torque sensor	lb/in^2

Equation 2 gives indicated stall speed (VSIKTS), knots:

$$VSIKTS = 0.5925 \left[\frac{2(w[\sin(ALPHA)(ax)+\cos(ALPHA)(az)] - LT-TSINAT)}{(po)\ (sw)\ (CLVCRT)}\right]^{1/2} \quad (2)$$

In these equations,

po = 0.002378 $slug/ft^3$ (air density at sea level)

w = current weight of aircraft, lb (calculated)

ALPHA = stall angle of attack, degrees (constant)

ax = longitudinal acceleration, g (measured)

az = vertical acceleration, g (measured)

LT = lift from the tail, lb (calculated)

TSINAT = vertical component of thrust, lb (calculated)

sw = wing area, square feet (constant)

Q = dynamic pressure (measured)

CLVCRT = critical effective coefficient of lift (a calculated polynomial).

These two equations represent only a small portion of the total calculations made by the microprocessor and its arithmetic coprocessor. Such calculations are performed up to one hundred times per second in order to keep the pilot informed of airspeed and stall speed. (Ref. 53). To derive these speeds, the system stores, within EEPROM, several coefficients that are aircraft- and flight-test-configuration dependent. The coefficients were measured on the initial flight test of the aircraft with the system onboard. (Ref. 53).

4.5.2 General Hardware Description

The Stall-Speed Warning System is enclosed in two main parts, one of which contains the signal conditioning, processing, I/O circuitry, along with the aural warning. The other part of the Stall-Speed Warning System contains the electronics used to drive the cockpit indicator. The part containing most of the electronics is called the Airborne Instrumentation Computer System (AICS). (Ref. 54). It represents NASA/Dryden's effort to standardize the design of the system for ongoing and future flight-test needs. The AICS provides a standard-size enclosure with a single-board microcomputer and a standard back-plane that accommodates commercially available or special-purpose, in-house-designed custom boards. The bus used in the AICS is the widely used "STD" bus developed by Pro-Log Corporation. (Ref. 55). The AICS contains six slots. As developed and configured for this system, five of the slots contain a mixture of commercially available and in-house designed boards:

1. Single-board processor
2. EEPROM board
3. Parallel-interface board
4. Analog-to-digital converter board
5. Voice-synthesizer and audio-power amplifier board

Figures 19 and 20 provide a system overview of the Stall-Speed Warning System and its processor board. Two boards (the single-board processor board and the voice-synthesizer amplifier board) were designed and built at NASA/Dryden; the remaining three boards were bought commercially. The commercial boards were modified to make them more environmentally hardened.

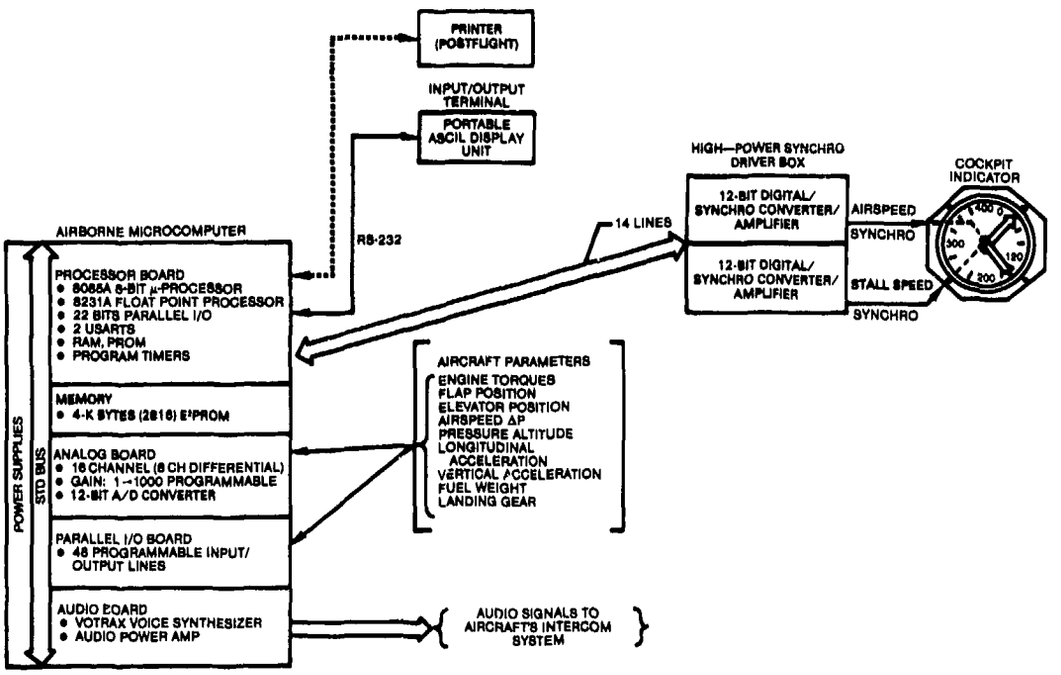


Figure 19. Stall-Speed Warning System.

The electronics part of the Stall-Speed Warning System contains the drive circuitry for the cockpit-mounted indicator as shown in Figure 21. This circuitry includes two separate digital-to-synchro converters with high-power transistor amplifiers. Each digital-to-synchro board is capable of driving any standard load (CT- or CDX- passive loads and torque receivers) of up to a 3-A peak. The indicator for visually displaying airspeed and stall speed is a modified, dual-needle radio magnetic indicator (RMI). NASA installed the same airspeed-indicator dial face as is used in the OV-1C aircraft. Along with the conventional airspeed pointer, another pointer with a red triangle was added to indicate calculated stall speed. The portable visual display unit, purchased commercially, interfaces the microcomputer board through a serial RS-232C port. The visual display unit inputs data and commands to the AICS and displays both calculated and raw-data values. Another RS-232C port allows post-flight dumping of test data in engineering units and provides derived coefficient of lift.

4.5.3 Single-Board Microcomputer

These are salient features of the single-board microcomputer:

1. 8085A microprocessor
2. 8231A floating-point processor
3. 22 bits of parallel I/O
4. Two USARTs that are programmable
5. 10.25K-bytes onboard memory configured as 10K-bytes/EPROM and 256-bytes/RAM or 8K-bytes/EPROM and 2.25K-bytes/RAM
6. Three 16-bit programmable timers
7. Address decoding
8. STD bus interface

The single-board microcomputer is a fully flight-qualified board, built on a six-layer printed circuit board (PCB). The board contains 24 integrated circuits, as well as an assortment of miscellaneous support circuitry.

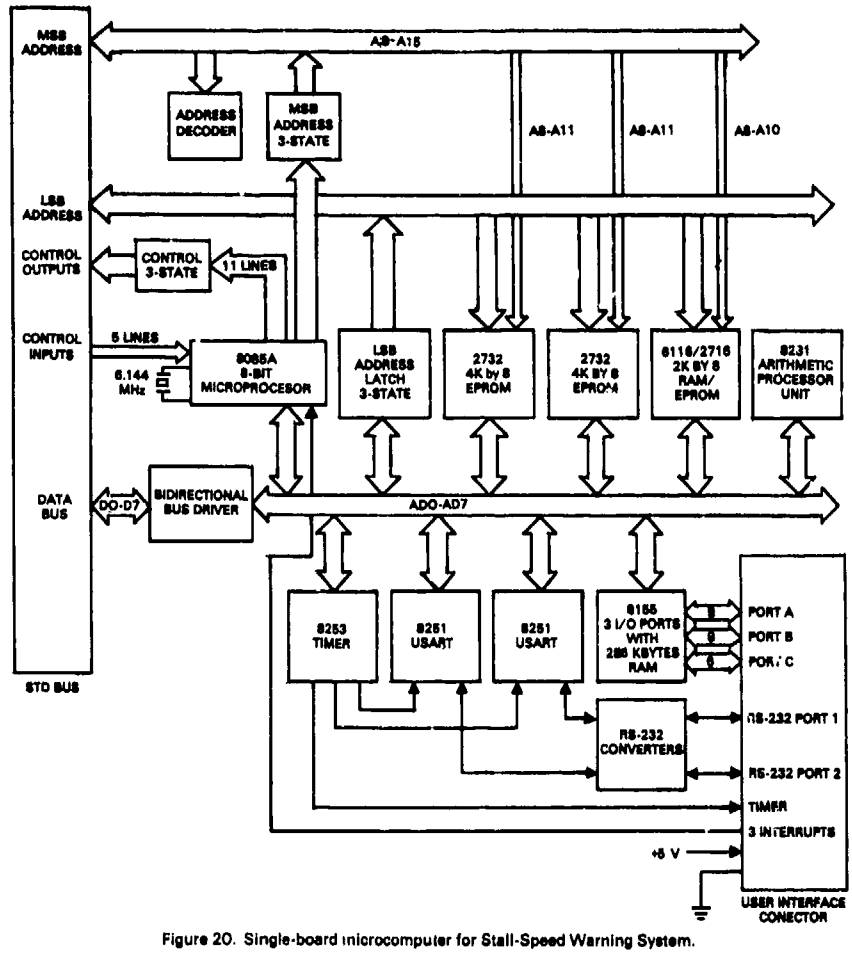


Figure 20. Single-board microcomputer for Stall-Speed Warning System.

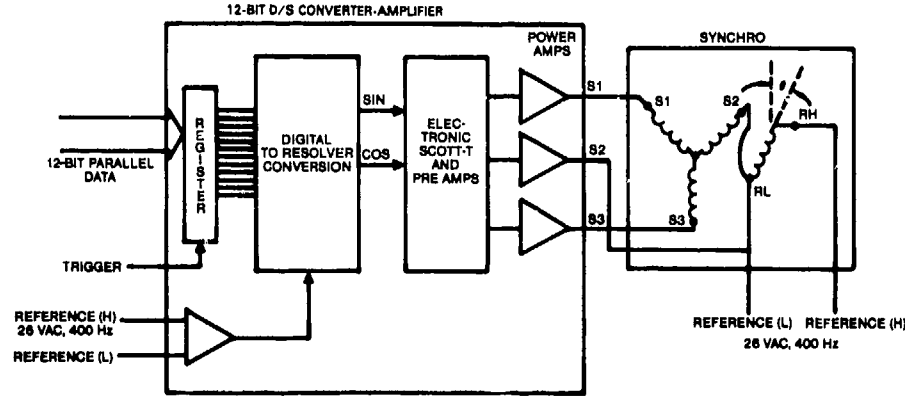


Figure 21. Digital-to-synchro driver for cockpit-mounted indicator.

The 8085A microprocessor has its lower 8 bits of address/data lines (AD0-AD7), as well as its control lines to control an 8155 integrated circuit. The 8155 provides two 8-bit and one 6-bit parallel I/O ports (used to interface the 8085A to the digital-to-synchro converters); 256 bytes of static RAM; and a 14-bit programmable timer for the board. Two 2732 EPROM-integrated circuits provide the board with 8K bytes of program memory. The remaining 2K bytes of RAM memory are provided by a Hitachi 6116 CMOS-integrated-circuit chip.

4.5.4 8231A Arithmetic Processor (Coprocessor)

The 8231A is interfaced to the 8085A through the lower 8 bits of the address/data bus (AD7-AD0), as well as through the control lines for read and write, clock, chip select, and the least significant address line, A0. The 8085A passes data and commands (which arithmetic operations to do) to the 8231A via the 8-bit data bus and then places the data into an internal stack. The 8231A can store eight single-precision, fixed-point values or four double-precision, fixed-point value words in an internal stack. The floating-point format uses a 32-bit word, with the least significant 24-bits representing the mantissa and the most significant 8 bits representing the exponent. Thus, the range of values represented in this way is $\pm(2.7 \times 10^{-20}$ to $9.2 \times 10^{18})$. With the 32-bit floating point and the 32-bit integer modes, this arithmetic processor has in excess of 40 special arithmetic instructions (Ref. 56).

4.5.5 EEPROM

EEPROM is needed to provide storage of coefficients and sensor data during the initial flight-test phase. This data must be retained even when the aircraft's power is off. After the flight, the data must be analyzed by the project engineer in the laboratory. EEPROM is commercially available using Intel 2816 nonvolatile EEPROM ICs. The board, when fully populated with eight 2816 chips allows for 16K bytes of memory. This application used only 2K bytes.

EEPROM is treated by the 8085A microprocessor as an offboard pseudo-RAM memory interfaced through the STD bus. Since the write operation to the board is longer than that for normal RAM, wait periods are required during which the 8085A does nothing. Also, due to the nature of the write electronics on the EEPROM board, a read operation immediately after a write operation is not recommended, since the write electronics take several microseconds to settle.

4.5.6 Voice Synthesizer

This function interfaces the processor board to a voice synthesizer and audio power amplifier via the STD bus. The board uses a Votrax SC-01 phonemic speech synthesizer that has a repertoire of 64 phonemes (addressable through a 6-bit data word). Two bits govern the pitch of the phoneme. Additional pitch control is obtained by using a 5497 binary-rate multiplier. An 8755 (2K bytes of EPROM and 16 I/O lines) is also on this board. The analog-synthesized "words" are then amplified with an adjustable-gain power amplifier capable of driving approximately 0.5 W into an 8-ohm speaker or of driving a higher impedance, audio-intercom system.

When the processor determines that the indicated airspeed is within a predetermined percentage of the stall speed, the processor outputs an aural warning to the pilot through this board. This predetermined airspeed threshold is set as the greater of 107.5 percent of indicated stall speed or 7.5 knots plus the indicated stall speed. These two values can be increased or decreased in real time through a terminal. Once amplified, the synthesized message is fed into the pilot's headset alerting the pilot to observe the airsperd. Also, an algorithm is used to increase the number of times this aural message is sent. The frequency is based on the difference between the two speed values, but the message is not sent more than 0.9 times a second.

Figures 22 through 25 detail various parts of the Stall-Speed Warning System.

4.5.7 System Software

The software for the Stall-Speed Warning System is made up of five major sections:

1. Initialization routine
2. Input-data formatting
3. Data collection
4. Data-processing and velocity calculations
5. Cockpit-indicator drive and aural output

During the first phase of flight testing, data is collected for predetermined flight conditions. Such data is time-tagged with an event marker actuated by the flight-test engineer when flight conditions are correct. At each time-tagged event, all parameters are loaded into the EEPROM circuits. During the flight, a maximum of 60 separate samples can be obtained (2K bytes of nonvolatile memory storage). Upon flight completion, the data can be outputted to a printer and analyzed on a larger computing system. The resulting output of the derived coefficient of lift makes it easy to spot-check data without the aid of a large computer. The purpose of the exercise is to determine the coefficients of the stall-speed equations. The coefficients can then be loaded back into the AICS via the visual display unit. A new flight is necessary to validate the stall-speed calculations. During this phase of flight testing, the visual display unit also displays the various parameters to the flight-test engineer. Indicated velocity, as well as indicated stall speed, are shown. However, the validity of the stall speed is only as accurate as the coefficients. Data collection takes approximately 10 to 54 microseconds for the entire set of parameters. This ensures an update of each parameter 20 times per second. Approximately 44 microseconds of the period are necessary for outputting to the display. The visual display itself is updated once every 0.5 seconds, while the cockpit indicator is updated every sample. Ten microseconds are necessary to update sensor calibrations and synchros. The entire operation is intended to collect data from the 11 sensors, to process the data, to perform velocity calculations, and to output results by display and synthesized voice to the pilot.

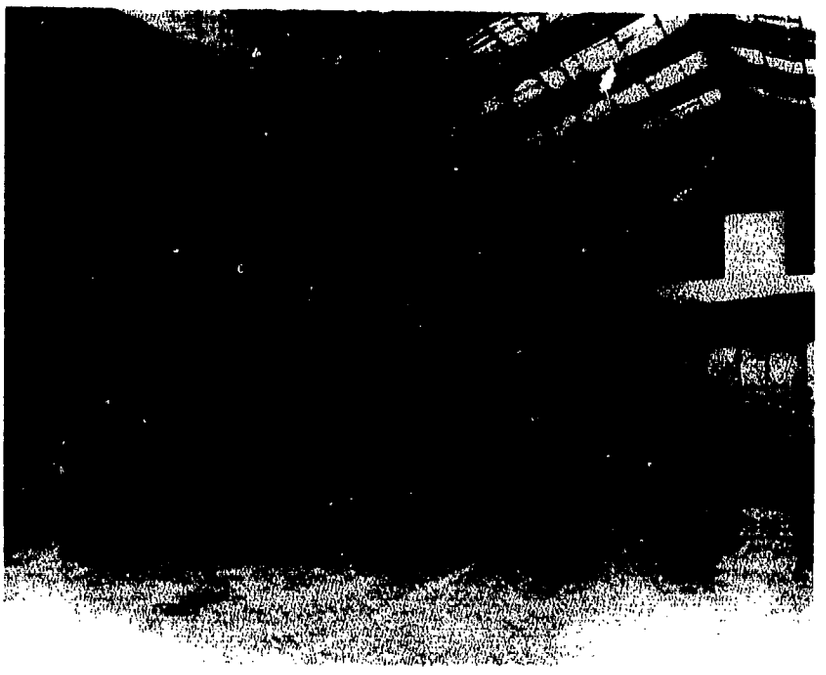


Figure 22. U.S. Army OV-IC Mohawk aircraft.

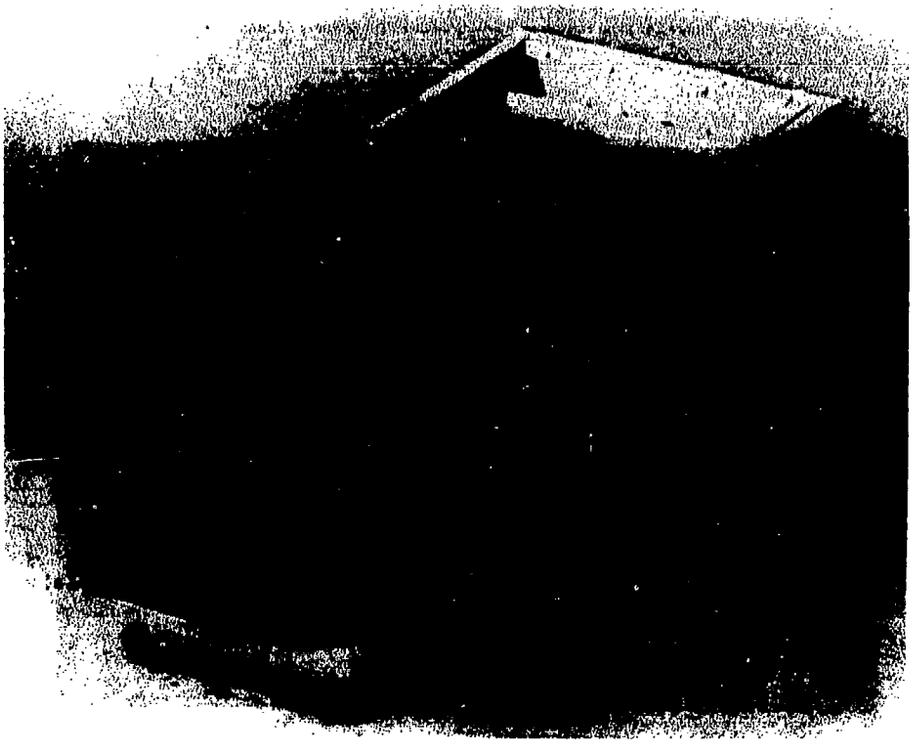


Figure 23. Prototype of the AICS, Stall-Speed Warning System.

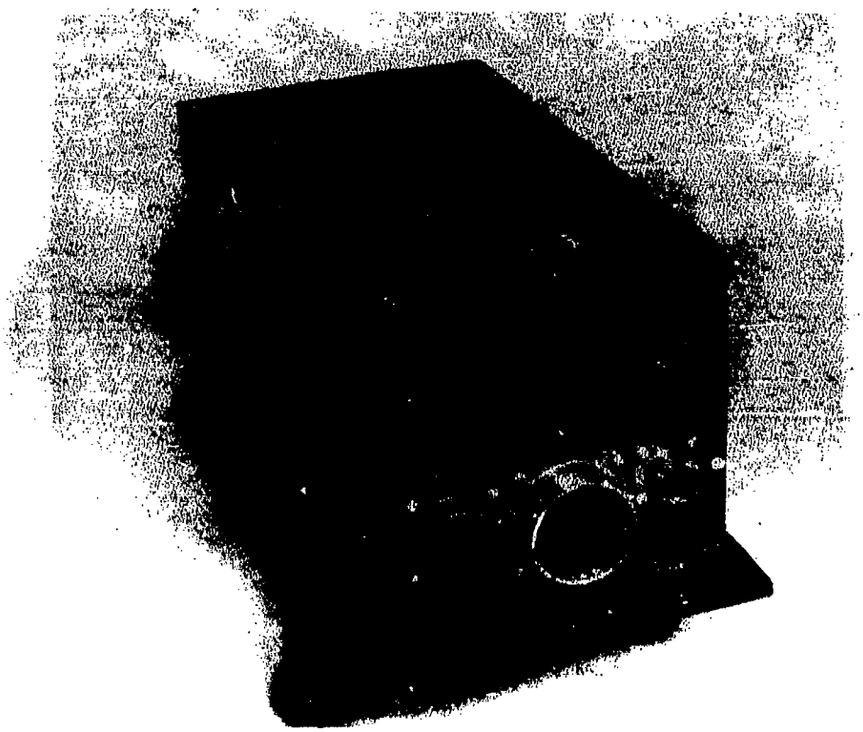


Figure 24. Final version of the AICS, Stall-Speed Warning System.

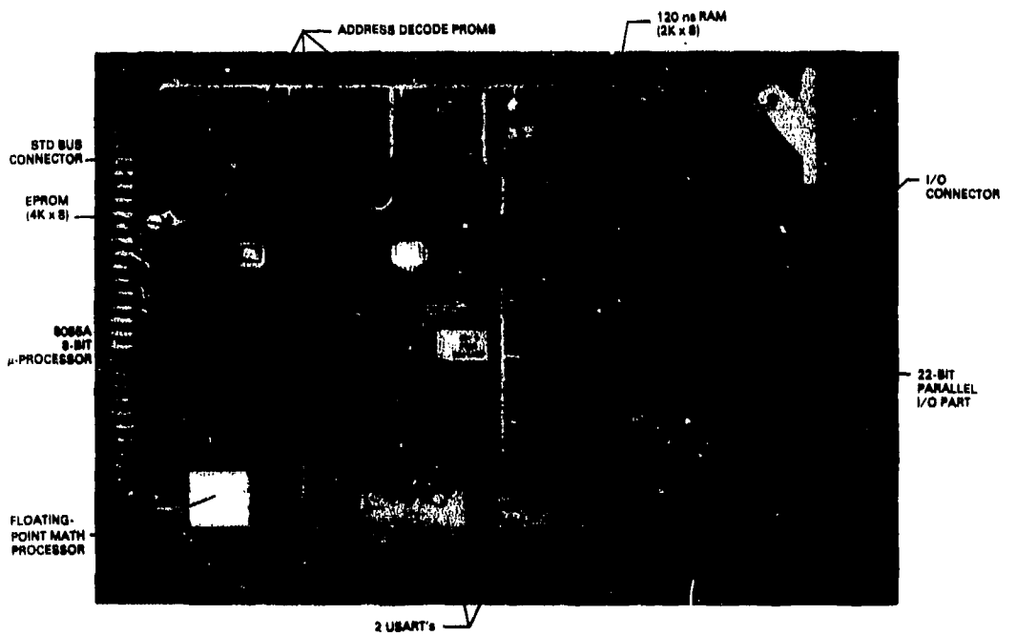


Figure 25. Processor board, Stall-Speed Warning System.

Technical Sources: Mr. Glenn A. Bever and Mr. Douglas O. Wilner, Senior Electronics Engineers, NASA, Ames Research Center, Dryden Flight Research Facility, Edwards, California. (Mr. Wilner is no longer a NASA employee).

4.6 ARINC-429 TELEMETRY INTERFACE (DFVLR, WEST GERMANY)

The DFVLR at Braunschweig, Federal Republic of Germany, has developed a microprocessor-based system that monitors, decodes, displays, and sends to a telemetry unit selected flight parameters on the ARINC aircraft bus. (Ref. 57). The platform is the DFVLR Advanced Technology Test Aircraft System (ATTAS). The ATTAS is an instrumented aircraft used for experimental development of avionics systems and in-flight simulation. The microprocessor is an 8-bit Rockwell 6502. (See Figure 26.)

The multipurpose system has these general functions:

1. Interfaces the ARINC-429 general aircraft bus to a DVR-182 database computer and PCM telemetry encoder made by the Hentschel Company.
2. Decodes and displays selected data in engineering units.
3. Stores and displays ARINC-429 input data as it is sent to the telemetry encoder.
4. Uses an operator keyboard to select data for display.
5. Presents selected data to as many as five parallel auxiliary outputs.
6. Offers the flight-test engineer several diagnostic self-test routines.

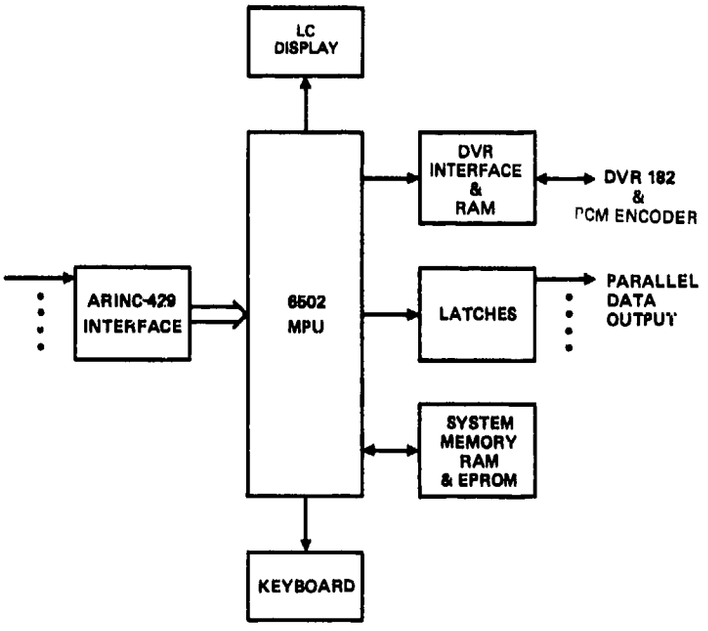


Figure 26. ARINC-429 telemetry interface.

The system input connects to the ARINC bus with shielded cable and outputs to a PCM telemetry encoder with 76 output data and two handshaking lines. The 24 bits of an ARINC data word are latched into a buffer register of the DVR-782 interface. This register accepts new data only after the complete contents have been transferred to the DVR-782. A 6522 Versatile Interface Adapter (VIA) buffers data to an auxiliary output.

An LC display is used with the operation keyboard to denote selected flight parameters. The 6502 microprocessor is used to convert the ARINC format to engineering units. For example, with the position of the ATTAS aircraft displayed, the operator can also, if necessary, display the flight parameters in a binary format. The system-hardware status check, general error messages, and operational command menus can also be displayed.

4.6.1 The Microprocessor

The 6502 is a popular 8-bit microprocessor. It has several peripheral integrated-circuit devices in its chip set. It can execute 56 basic instructions with 13 address modes. It can directly address 65,536 memory locations, and it is available in a low-power CMOS version.

4.6.2 Additional Airborne Microprocessor-based Systems

DFVLR, Braunschweig, has developed two other microprocessor-based flight-test systems: an in-flight antenna calibration unit and an IRIG-time-code reader. Both systems were developed using the Rockwell 6502 microprocessor as the control unit. (Refs. 58, 59).

Technical Sources: Joerg Wolf, Joachim Gabriel, Kurt Klein, Test Instrumentation Design Engineers, DFVLR, Braunschweig, Federal Republic of Germany.

5. COMMERCIAL MICROPROCESSORS AND SUPPORT DEVICES

Microprocessors are commercially available with word lengths of 4, 8, 16, and 32 bits. Table 5 indicates the general performance level an instrumentation-design engineer can expect with typical microprocessor devices. Electronics trade magazines provide information on current microprocessor and support devices. EDN magazine has been publishing microprocessor performance data and related support-device directories on an annual basis for several years. (Refs. 60, 61). The information in Tables 5 and 6 was obtained from Refs. 60 through 64 and represents a small portion of the total information available on microprocessor devices.

Table 5. Comparison of selected microprocessors (June 1985).

Part[1]	Manufacturer	Type	Word Bit Width	Address Bus Width	Speed	Clock	Date
2901	Advanced Micro Devices	Bit Slice	4*	Varies	0.115	10.0	1974
29300	Advanced Micro Devices	Bit Slice	32*	32	0.08	12.5	1985
6502	Commodore	MPU	8	16	3.0	4.0	1975
F8	Fairchild	MPU (3850)	8	N/A	2.0	2.0	1974
F9450	Fairchild	MPU (MIL-STD-1750A ISA)	16	16	0.2	20.0	1983
T424	INMOS	Transputer	32	32	0.1	5.0	1985
8048	Intel	Microcomputer chip[2]	8	N/A	1.4	8.0	1977
8080	Intel	MPU	8	16	2.0	2.0	1974
8085	Intel	MPU	8	16	1.3	3.0	1976
8086	Intel	MPU	16	20	0.6	5.0	1978
8088	Intel	MPU	8	20	0.6	5.0	1979
80286	Intel	MPU	16	24	0.25	8.0	1984
80386	Intel	MPU	32	32	0.28	20.0	1985
6800	Motorola	MPU	8	16	1.0	2.0	1974
6805	Motorola	Microcomputer Chip	8	N/A	2.0	4.0	1979
6809	Motorola	MPU	8	16	1.5	2.0	1979
68000	Motorola	MPU	16	24	0.37	12.0	1980
68020	Motorola	MPU	32	32	0.3	16.0	1984
1802	RCA	MPU	8	16	5.0	3.2	1978
PPS-4	Rockwell	Microcomputer Chip	4	N/A	8.3	0.2	1974
TMS1000	Texas Instruments	Microcomputer Chip	4	N/A	10.0	0.4	1974
TMS9900	Texas Instruments	MPU	16	15	2.0	3.0	1976
Z8	Zilog	Microcomputer Chip	8	N/A	1.5	8.0	1976
Z80	Zilog	MPU	8	16	0.7	6.0	1976
Z8000	Zilog	MPU	16	24	0.4	10.0	1981
Z80,000	Zilog	MPU	32	32	0.2	25.0	1985

NOTES:

1 Each part number can include several variations.
2 Microcomputer chip = MPU + memory + special I/O on one chip.
* Word size/device slice.
Clock = MPU clock frequency in megahertz (MHz).
Speed = shortest instruction-execution approximate time, in microseconds.
Date = approximate date of introduction.

Table 6 provides a representative sample of the special-purpose support devices used in microprocessor chip sets. The available support devices sometimes determine what microprocessors are selected.

The transputer is a new microprocessor device developed by INMOS. Although the device has probably not been used in an airborne flight-test application, it may be used in the future whenever system speed and concurrent processing is required. The INMOS T424 transputer is essentially a 32-bit microprocessor with these features: on-chip memory (4-K RAM); addressing capability up to 4-G bits; and serial link interfaces. Its instruction set was designed for use with high-level languages. The transputer can execute instructions in 100 nanoseconds. It can function as a single processor, or it can be linked to other transputers to do parallel processing tasks.

Table 6. Selected microprocessor special-purpose support devices (June 1985).

Function	Manufacturer	Part No.	Microprocessor Compatibility	Comments
Parallel I/O	Harris	82C55A	8080/8086	5 MHz, CMOS
Parallel I/o	Motorola	68230	68000	10 MHz, NMOS
Parallel I/O	NEC	7210	8086	8 MHz, IEEE-488
Serial I/O	Signetics	68562	68000	4M baud, NMOS
Serial I/O	Zilog	Z84C40	Z80	800K baud, CMOS
System Clock, Timer	Hitachi	HD6340	6800	2 MHz, CMOS 3 16-bit Timers
System Clock, Timer	Zilog	Z84C30	Z80	4 MHz, CMOS 4 16-bit, Timers
Arithmetic Processor	Motorola	68881	68000	16 MHz, Trig IEEE Floating Point
Arithmetic Processing	Intel	8231A	8085A	2 MHz, Trig 32-bit Floating Point
Interrupt Controller	Intel/Harris	825C59A	8080/85 8086/88	CMOS, 8 Priority Levels
DMA Controller	Motorola	68442	68000	10 MHz, NMOS
Disk Controller	Signetics	68454	68000	10 MHz, NMOS hard and floppy
CRT Controller	Motorola	68486	68000	14 MHz, HCMOS
LAN* Controller	Intel	82586	8086/88 80286	NMOS Ethernet, IBM PC
Analog Interface	Texas Instruments	TLO808	general	Successive approximation 8 channels, 8-bit A/D
Speech: Analysis/ Synthesis	OKI	MSM 5128	general	0-64K bps CMOS

* LAN = Local Area Network

6. REFERENCES

1. "The Integrated Circuit ERA (1959-1975)," Electronic Design, Vol. 24, No. 4, February 16, 1976.
2. MC 68020, Motorola, 1984.
3. Clay, C. W., "Digital Electronic Flight Decks: The Outlook for Commercial Aviation," AES 20, No. 2, March 1984, pp 221-226.
4. Spitzer, Cary R. "The All-Electric Aircraft. A Systems View and Proposed NASA Research Programs," AES 20, No. 2, March 1984, pp 261-265.
5. Moore, Gordon E., "Progress in Digital Integrated Electronics", IEEE International Electron Device Meeting, December 1975, pp 11-13.
6. Barbe, D. F. and E. C. Urban, "VHSIC Technology and Systems," Very Large Scale Integration: VLSI Fundamentals and Applications, IEEE Proceedings, 1982, pp 255-271.
7. Gibson, W. M., "Flight Testing DeHavilland Dash-8 Aircraft Utilizing On-board Data Analysis by Microprocessor (TRS-80)," Proceedings of the AIAA Flight Testing Conference," Las Vegas, NV, November 1981.
8. Meany J. J. "The Evolution of Flutter Excitation at McDonnell Aircraft," Proceedings of the 14th Annual Society of Flight Test Engineers Symposium, Newport Beach, CA, August 1983.
9. Tabb, J. A. "Lockheed Airborne Data System: Distributed Microprocessors Provide Onboard Real-Time Analysis," Proceedings of the AIAA Flight Testing Conference, AIAA-81-2367, Las Vegas, NV, November 1981.
10. Veatch, D. W. and R. K. Bogue, "Analogue Signal Conditioning for Flight Test Instrumentation," AGARDograph No. 160, Vol. 17, April 1986.

11. McSharry, M. E., "Fault Tolerant Flight Control Avionics Integration Using MIL-STD-1533B," Proceedings of the IEEE/AIAA 5th Digital Avionics Systems Conference, paper 11.1, October 1983.

12. Burke, M. J. and E. W. Ferris, "Application of a Microprocessor-Controlled Cockpit Display for Enhanced Pilot Control of Flight-Test Maneuvers," Proceedings of the AIAA Flight Testing Conference, AIAA-81-2510, Las Vegas, NV, November 1981.

13. Wood, I. C. et al., "Design Techniques for a 565/680 Mbit/s Coder/ Decoder," IEE Proceedings, Vol. 132, Pts E and I, No. 2, March/April 1985, pp 68-72.

14. Aeronautical Radio, Inc., ARINC Specifications 429-7, Mark 33 Digital Information Transfer System (DITS), Annapolis, MD, revised January 3, 1983.

15. Aircraft Internal Time Division Command/Response Multiplex Data Bus, DoD MIL-STD-1553B, September 21, 1978.

16. Special Centennial Issue on Standards, IEEE Micro, Vol. 4, No. 4, August 1984.

17. Bailey, Chris, "Bus Standards Enrich Microcomputer Board Variety," Electronic Design, April 15, 1982.

18. Javetski, John, 32-Bit Microsystem Buses Gear Up for the Next Frontier, Electronic Products, July 2, 1984.

19. Millman, Jacob, "Microelectronics, Digital and Analog Circuit and Systems," McGraw-Hill, 1979, Chs. 1, 2, 3, 4, 8.

20. Mead, Carver and Lynn Conway, "Introduction to VLSI Systems," Addison-Wesley, 1980, Chs. 1,2.

21. Rauscher, Tom and G. Linson, "A United Approach to Microprocessor Software Development," IEEE Computer, Vol. 11, No. 6, June 1978, pp 44-54.

22. Prickett, Michael J., "Microprocessors for Dedicated Control and Instrumentation Applications," Proceedings of the 28th Annual International Instrumentation Symposium, Instrument Society of America, May 1982.

23. Leventhal, Lance A., "Introduction to Microprocessor Software, Hardware, Programming," Prentice-Hall, 1978, Ch. 6.

24. Howes, Norman R., "Managing Software Development Projects for Maximum Productivity," IEEE Transactions on Software Engineering, Vol. SE-10, No. 1, January 1984.

25. Noble, William B., "Developing Safe Software for Critical Airborne Applications," Proceedings of the IEEE/AIAA 6th Digital Avionics Systems Conference, AIAA-84-2598 CP, December 1984.

26. Hatley, D. J., "The Use of Structured Methods in the Development of Large Software-Based Avionics Systems," Proceedings of the IEEE/AIAA 6th Digital Avionics Systems Conference, AIAA-84-2595 CP, December 1984.

27. Buckley, Fletcher J., "The IEEE Software Engineering Standards Process," IEEE Micro, August 1984, pp 57-62.

28. "Military Standard Software Development," U.S. DoD MIL-STD-1679A, October 22, 1983.

29. "Software Considerations in Airborne Systems and Equipment Certification," Radio Technical Commission for Aeronautics (RTCA), Document No. RTCA/DO-178A, March 1985.

30. Borek, R. W. "Practical Aspects of Instrumentation System Installation," AGARDograph No. 160, Vol. 13, September 1981, pp 2-18.

31. "Electronic Equipment, Airborne General Specifications," U.S. DoD MIL-E-5400T, November 16, 1979.

32. "Environmental Test Methods," U.S. DoD MIL-STD-810C, March 10, 1975.

33. "Climatic Extremes for Military Equipment," U.S. DoD MIL-STD-210B, December 15, 1973.

34. "Environmental Conditions and Test Procedures for Airborne Equipment," Radio Technical Commission for Aeronautics (RTCA), Document No. RTCA/DO-160B, July 1984.

35. Coppola, Anthony, "Reliability Engineering of Electronic Equipment, A Historical Perspective," IEEE Transactions on Reliability, Vol. R-33, No. 1, April 1984, pp 29-35.

36. O'Connor, P. D. T, "Microelectronic System Reliability Prediction," IEEE Transactions on Reliability, Vol. R-32, No. 1, April 1983.

37. "Test Methods and Procedures for Microelectronics," U.S. DoD MIL-STD-883C, August 25, 1983.

38. "General Specification for Microcircuits," U.S. DoD MIL-M-38510F, October 31, 1983.

39. "Reliability Prediction of Electronic Equipment," U.S. DoD MIL-HDBK-217D, Notice 1, June 13, 1983.

40. Intel Component Quality/Reliability Handbook, 1985, p 35.

41. Doyle, Edgar A., "How Parts Fail," IEEE Spectrum (A Special Issue on Reliability), Vol. 18, No. 10, October 1981, pp 36-43.

42. Kaye, Harvey, "Junction-Temperature Planning Boosts for Product Reliability," EDN, April 18, 1985.

43. Motorola MOS Microprocessor Group Reliability and Quality Monitor Results, 1st Quarter 1985.

44. Fung, T. C., An Introduction to the Theory of Aeroelasticity, Galcit Aeronautical Series, John Wiley and Sons, 1955.

45. Scanlan, R. H. and R. Rosenbaum, Introduction to the Study of Aircraft Vibration and Flutter, The Macmillan Company, 1960.

46. Zimmerman, N. H. and J. T. Weissenburger, "Prediction of Flutter Onset Speed Based on Flight Flutter Testing at Subcritical Speeds," Journal of Aircraft, Vol. 1, No. 4, 1964.

47. Van Nunen, J. W. G. and G. Piazzoli, "Aeroelastic Flight Test Techniques and Instrumentation," AGARDograph No. 160, Vol. 9, 1979.

48. Wright, A., "Product Description of the Tornado IDS AFDS Failure Simulation System," AAEE File Reference, ACI/150/10, 10 August 1983.

49. Stanfield, T., "Specification for the Design and Build of an In-Flight Failure Simulation Box for the IDS Tornado AFDS," AEN/29/705, 11 June 1982.

50. Wright, A., "The Tornado In-Flight Failure Simulation Box - A Discussion Paper," CIG Instrumentation Note, No. 3/82, 23 June 1982.

51. "The Dual ARINC Display Unit," NLR Memorandum IE-83-040U, August 1983. (in Dutch)

52. Wilner, Douglass O. and Glenn A. Bever, "An Automated Stall-Speed Warning System," NASA Technical Memorandum 84917, February 1984.

53. Underwood, D. L. and R. S. Adler, "Preliminary Airworthiness Evaluation of a National Aeronautics and Space Administration Automated Stall Warning System for an OV-1 Aircraft," U.S Army Aviation Engineering Flight Activity, Edwards Air Force Base, CA 93523, USAAEFA Project No. 81-06-1, Final Report, July 1984.

54. Bever, Glenn A. "The Development of an Airborne Instrumentation Computer System for Flight Test," Proceedings of the Symposium on Flight Test Techniques, Flight Mechanics Panel, AGARD-CP-373, Lisbon, Portugal, April 1984.

55. "STD Bus Technical Manual and Product Catalog," Pro-Log Corporation, Monterey, CA, August 1982, pp 1-1 to 2-5.

56. "Microprocessor and Peripherals Handbook," Intel Corporation, Santa Clara, CA, 1983.

57. Wolf, Joerg, "ARINC 429 Telemetry Interface," Report No. IB 112-84/26, DFVLR, 1984. (in German)

58. Gabriel, J., "A Position-Finding Aircraft Equipment for the In-Flight Antenna Calibration Which Uses Several TACAN Ground Facilities," Report No. IB 112-81/09, DFVLR, 1981. (in German)

59. Klein, K., "Small IRIG-Timecode Reader Based on Microprocessor Techniques," Report No. IB 112-82/13, DFVLR, 1982. (in German)

60. Cushman, Robert H., "Eleventh Annual Microprocessor/Microcomputer Chip Directory," EDN, Vol. 29, No. 23, November 15, 1984.

61. Cushman, Robert H., "Microprocessor Support Chips Present a Wide Array of Choices," EDN, Vol. 30, No. 25, March 7, 1985.

62. MacGregor, Doug, Dave Mothersole, and Bill Moyer, "The Motorola MC68020," IEEE MICRO, August 1984, pp 101-118.

63. Javetski, John, "32-Bit Processors Pack Mainframe Muscle," Electronic Products, July 16, 1984, pp 49-55.

64. 1985 IC Master, Hearst Business Communications, Inc., Santa Clara, CA.

ACKNOWLEDGMENT

The author wishes to thank the several flight-test engineers who provided the primary application material, and who reviewed the many drafts of this document. Their comments and suggestions were very much appreciated. The sources for the selected, detailed application information in Section 4 have been shown at the end of each application.

APPENDIX A

A LIMITED GLOSSARY OF MICROPROCESSOR/MICROCOMPUTER TERMS

Access Time. The delay between the time when a memory receives an address and the time when the data from that location is available at the output.

Accumulator. A register that is used to accumulate or collect results. For example, a computer performs addition by adding a number in memory to the contents of an accumulator; the sum remains in the accumulator.

Active - High. The active logic state is the one state.

Active - Low. The active logic state is the zero state.

ADA. DoD standard language (MIL-STD-1815A) intended to replace all existing DoD languages used for embedded computer systems.

Address. The identification code that distinguishes one location of memory from another. The CPU can use the address to select a particular location.

Address bus. A bus that the CPU uses to select a particular location of memory or an input/output port for a data transfer.

Addressing modes. The means to specify the memory addresses that the computer needs to execute an instruction. These addresses may tell the computer where to obtain data, where to store data, or where to find the next instruction.

Algorithm. A well-defined set of rules or procedures for the solution of a program.

Analog. Continuous signal or quantity that can take any value.

Applications software. The programs that actually do the work (e.g., print lists, convert flight parameters, or monitor a temperature) as opposed to programs that help operate the computer or help programmers write other programs (systems software).

Architecture. Structure of a system; in the case of computers, architecture often refers specifically to the way in which the CPU is organized.

Arithmetic Logic Unit (ALU). A subsystem that can perform a variety of arithmetic or logical functions under the control of function inputs.

ASCII. American Standard Code for Information Interchange; a 7-bit code often used to represent typed characters in computers.

Assembler. A computer program that converts assembly language programs (source code) into the numerical form (object code) that the computer actually executes. The assembly converts mnemonic operation codes (such as ADD, SUB, or MOVE) into their numerical equivalents, replaces symbols or labels with their numerical equivalents, and assigns locations in the computer's memory to data and instructions.

Assembly language. A programming language (specific to a particular CPU) in which the programmer uses mnemonic operation codes, names, and labels to refer directly to their numerical equivalents. Assembly language is a low-level language, since the assembler translates each statement directly into one machine-language instruction (rather than into a series of machine- language instructions, as is the case with a high-level language). Writing assembly-language programs requires detailed understanding of a particular CPU.

Asynchronous. Operating without reference to an overall timing source.

BASIC. Beginners' All-purpose Symbolic Instruction Code; a widely used interactive computer language that is especially well-suited to personal computers and beginning users. BASIC was developed by Kemeny and Kiutz in the middle 1960s at Dartmouth College.

Baud. A measure of the rate at which data is transmitted, expressed in terms of the number of state changes per second. Common data rates are 300, 1200, 2400, 4800, and 9600 baud.

BCD. Binary-coded decimal; a method for representing decimal numbers in which each decimal digit is coded in four bits.

Benchmark. A problem or program that can be used as a basis for comparing computers, languages, etc.

Bidirectional. Able to transfer data in either direction.

Binary. Number system with base 2; the only digits are 0 and 1. Also refers to any system that has only two possible states or levels, such as a switch that is either on or off.

Bit. A binary digit; it can only have the values 0 or 1.

Bit Slice. A section of CPU that may be combined in parallel with other such sections to form a complete CPU with various word lengths.

Block. A set of data or memory locations.

Bootstrap (loader). A program that starts the computer and prepares it to load other programs into its memory. The bootstrap may reside in read-only memory or the operator may have to enter it by hand.

Bottom-up design. A method for designing programs in which the programmer designs, codes, and tests all the parts of the program (or modules) before combining them.

Branch instruction. An instruction that specifically tells the microprocessor/microcomputer where to get the next instruction, thus forcing it to depart from its normal sequential execution. Branch instructions explictly change the program counter. They may be conditional, that is, only tell the computer where to get the next instruction if a condition is satisfied.

Breakpoint. A condition under which the computer is to stop execution of its current program; used as an aid in debugging programs. The programmer must specify (set) the breakpoints.

Buffer. A temporary storage area for data; often used to hold data that has just been read from an input device or is about to be sent to an output device.

Bus. A group of parallel wires that allow data transfer between functional devices. Examples: data bus, control bus, address bus.

Bus contention. A situation in which two or more devices are trying to place data on a common bus at the same time.

Byte. The basic grouping of bits that a computer can handle at one time, 8 bits in length. One byte can hold one type ASCII character or two decimal (BCD) digits.

C. A high-level programming language specifically designed for writing systems software; originally developed at Bell Telephone Laboratories.

Chip. A substrate containing a single integrated crcuit.

Clock. A regular timing signal (i.e., a series of pulses equally spaced in time that tell a computer or a component when to do various functions.

CMOS. Complementary Metal Oxide Semiconductor; a type of integrated circuit that requires very little power and will withstand a large amount of electrical noise, often used in spacecraft and in portable equipment that must operate from a battery.

Coding. The writing of programs in a language that a computer can either execute or translate into an executable form.

Comment. A part of a computer program that explains what is happening but does not affect how the computer executes the program. Comments are used to document programs.

Compiler. A computer program that translates another program written in a high-level language into a program that a computer can execute. A compiler translates the entire program and produces an executable program (object file).

CPU. Central Processing Unit; the main control function of a microcomputer. Microprocessors are used as a CPU.

Data bus. A bus that is used to transfer data between the CPU and the memory or the input/output section.

Data file. A file that contains data rather than programs or procedures.

Debugger. A program that helps the user find and correct errors in other programs. The errors are called bugs.

Decoder. A device that converts coded information into its actual meaning.

Direct Memory Access (DMA). A method for quickly transferring data to or from a computer's memory without executing the main program.

Disassembler. A program that translates a machine-language program (object code) back into assembly language. A disassembler is the opposite of an assembler. It translates numbers into the mnemonic operation codes and names that the programmer can understand more easily.

Disk. A circular device with a magnetic surface that is used to record data. Floppy (flexible) disks are capable of holding hundreds of thousands of bytes, whereas hard disks are capable of holding tens of millions of bytes.

Disk controller. A device that controls the operations of a disk, the interface between a disk and a computer.

Diskette. An individual floppy disk, the recording medium in a floppy disk system. There is sometimes confusion among the terms diskette, disk, and floppy disk, but diskette refers only to the medium, not to the controller.

Disk Operating System (DOS). An operating system that allows the user to transfer programs and data to or from a disk. There are many different (usually incompatible) disk operating systems.

Documentation. The techniques used to describe a computer program so that it can be used, maintained, and updated.

Double-density. Special recording method for floppy disks that allows them to store twice as much data as in normal or single-density.

Double-sided. Recording on both sides of a floppy disk.

Dynamic memory. A memory that loses its contents gradually without any external causes. The contents simply leak away unless the computer periodically rewrites them into the memory; the rewriting is called refresh.

Editor. A computer program that lets the user prepare printed matter (text material) or input for other programs. It allows the user to correct, add, delete, or rearrange material.

Emitter-Coupled Logic (ECL). A high-speed bipolar technology for integrated circuits.

Emulation. To imitate one system with another.

EPROM (Erasable PROM). A Programmable Read Only Memory (PROM) that can be erased by exposing it to ultraviolet light. Complete erasure usually takes 20 to 30 minutes. (See PROM.)

EEPROM. Electrically Erasable PROM. Basically, a read-only-memory integrated circuit that can be erased with an electronic pulse and reprogrammed while in the circuit.

Executable. Can be executed by a computer without translation.

File. A collection of related records or data treated as a unit. Editors, assemblers, and compilers normally work on program or data files. System commands that the user needs repeatedly may be kept in a procedure file to save typing.

Firmware. Programs stored in read-only memory.

Flag. An indicator that is either on (1) or off (0) and can be used to select between two alternative courses of action.

Floppy disk. A flexible disk with a magnetic surface that can be used to store data. The surface is divided into areas called sectors. An individual floppy disk is often called a diskette.

Flowchart. A graphic representation of a computer program.

FORTH. A high-level programming language. Special-purpose commands can be defined and added easily. It is an interpreted language that has a relatively fast execution time.

General-Purpose Interface Bus (GPIB). IEEE-488 Bus. A standard interface intended for use in networks of instruments.

Global. Defined in more than one section of a computer program; a universal variable rather than one used only locally.

Hard disk. A disk that is not flexible; more expensive than a flexible (floppy) disk but capable of storing much more data.

Hexadecimal. Number system with base 16; uses the decimal digits (0 through 9) and the letters A through F as its digits.

High-Level Language (HLL). A programming language in which the statements represent procedures rather than single computer instructions. A compiler or interpreter translates a program written in a high-level language into a form that the computer can execute. Common high-level languages are BASIC, COBOL, FORTRAN, PASCAL, "C", ADA, JOVIAL, and FORTH.

Input/Output (I/O) section. The section of the computer that connects the CPU to the peripherals.

Instruction. An instruction tells the computer what operation to perform next.

Integrated Circuit (IC). A complete electrical circuit on a single semiconductor chip.

Intelligent (or smart) terminal. A terminal that has its own computing capability and can be used to process data even when it is not connected to a computer.

Interpreter. A computer program that executes programs written in a high- level language. An interpreter executes each statement immediately after reading it; it does not produce an object program, as a compiler does, so the program must be translated each time it is run.

Interrupt. An input to a computer that forces it to suspend its current program and execute a special program (or service routine). Interrupts are often used to tell the computer that peripherals are ready to send or receive data.

Interrupt service routine. A program that performs the actions required to respond to an interrupt.

I/O driver. A computer program that transfers data to or from an I/O device, also called an I/O utility.

I/O utility. A computer program that transfers data to or from an I/O device, also called an I/O driver.

Jovial. A high-level programming language. Primarily used for airborne embedded microprocessors and computers for DoD projects.

Jump instruction. An instruction that specifically tells the computer where to get the next instruction; similar to a branch instruction.

K. A term used to mean 1024 (2^{10}) when referring to the size of a computer memory.

Kilobit. 1024 bits, also referred to as 1K.

Kilobyte. 1024 bytes, also referred to as 1K.

Label. A name attached to a statement in a program so that other statements can transfer control to that statement or can use the name as if it were a memory address. The label takes the value of the memory address in which the executable version of the statement starts.

Large-Scale Integration (LSI). Integrated circuits of very great complexity on a single chip. Typical LSI circuits are memory chips, microprocessors, microcomputers, and calculator chips.

Latch. A device that retains its contents until new data is specifically entered into it.

Line editor. An editor that manipulates text as a series of lines rather than as a continuous string of characters.

Line printer. A printer that prints an entire line of characters at a time.

Linking loader. A program that combines a series of programs and subroutines that may have been assembled or compiled separately and places them all in memory with the required interconnections.

Loader. A program that reads another program from an input device into memory.

Logic analyzer. An electronic test instrument that can display the states of many time-varying signals at once.

Loop. A self-contained sequence of statements that the computer repeats until a terminating condition is satisfied.

Machine independent (or computer independent). Will run on any computer or on any computer that has a compiler or interpreter for a particular language. Also referred to as portable.

Machine language. A computer language that consists of instructions written in a numerical form that a computer can execute directly. Also called machine code or object code.

Macro. A name given to a series of assembly language instructions. The assembler replaces each reference to the macro with a copy of the series of instructions, thus saving a large amount of repetitive typing.

Maintenance (of programs). Updating and correcting a computer program that is in actual use.

Megabyte. One million bytes or 1,048,576 bytes (2^{20}).

Memory-mapped input/output. A method for addressing I/O ports whereby they are treated as if they were memory locations.

Memory protect. A feature that stops programs from writing into all or part of memory, thus preserving programs or data from accidental or unauthorized overwriting.

Microprocessor. A single-chip central processing unit on an LSI chip; common versions are the Intel 8080, 8085, and 8086; Commodore 6502; Motorola 6800, 68000; Zilog Z-80 and Zilog Z-8000.

Mnemonic. A name that suggests the actual meaning or purpose of the object to which it refers. Mnemonics are used to represent the microprocessor/computer instruction set.

Modular programming. A programming method in which the programmer divides the entire task into logically separate sections (or modules).

Monitor (software). A simple operating system that allows the user to enter programs and data into memory, run programs, and observe the contents of registers and memory locations. Operating systems can also have monitors that check tasks and users on systems.

MOS. Metal Oxide Semiconductor; a popular method of integrated circuit fabrication. Microprocessors are typically made using this technology.

MPU. Microprocessor Unit; a term used for the microprocessor device that is being used as the central processor of a system.

Multitasking. Executing several tasks at the same time without having to complete one before starting another. Each task may be given a slice of time or may be executed until its further operation requires the execution of another task or some external action.

Multi-user. Capable of handling more than one user at a time.

Nanosecond. 10^{-9} second, abbreviated ns.

Nested subroutines. Subroutines that are used inside other subroutines. The nesting level is the number of transfers of control that are required to reach a particular subroutine without returning to the main program.

Nonvolatile memory. A memory that does not lose its contents when power is removed. ROM, PROM, EPROM, and EEPROM are examples of nonvolatile memory.

Object file. A file containing instructions that the computer can execute directly.

Octal. Number system with base 8. Use decimal digits 0 through 7.

Operating system. A computer program that controls the overall operation of a computer and performs such tasks as assigning places in memory to programs and data, processing interrupts, scheduling jobs, and controlling the overall input/output system.

Operating code (or op code). The part of an instruction that tells the computer which operation to perform.

Parity. A code used to detect recording or transmission errors. Parity is a 1-bit code that makes the total number of one bits in a unit of data (including the parity bit itself) odd (odd parity) or even (even parity).

PASCAL. A popular high-level language that was developed specifically for use with structured programming and top-down design.

Pointer. A storage place that contains the address of a data item rather than the item itself. That is, a pointer tells where the data is located.

Procedure file. A file that contains procedural instructions for a computer program, (e.g., assignments required by an operating system or some other system program). A procedures file may, for example, contain all the commands required to assemble or compile a program under a particular system.

Program counter. A register that contains the address of the next instruction in memory the computer will execute.

PROM. Programmable Read-Only Memory; a memory that cannot be changed during normal computer operation but can be programmed under special conditions. Some PROMs (called EPROMs) can be erased with ultraviolet light and reused.

PROM programmer (or PROM burner). Special equipment that is used to change the contents of a PROM or EPROM.

RAM. Random-Access Memory; memory that can be both read and altered (written into) during normal operation.

Random-access. Referring to a storage device from which any data item can be removed in the same amount of time. Semiconductor main memory is random-access, whereas a tape is not.

Real-time clock. A device that interrupts a CPU on a regular basis, such as once per second. By counting the number of such interrupts, the computer can keep track of elapsed time.

Records. A collection of related data items. Records usually consist of several fields; in turn, a file usually contains several records.

Refresh. The process of restoring the contents of a dynamic memory before it is lost.

Register. A storage location used to hold data inside the CPU.

Relative branch. A branch instruction that causes the CPU to resume program execution after skipping over a specific number of memory locations. The specified number, called a relative offset, may be negative or positive.

Relocatable. Can be placed in any part of memory without changes; can occupy any set of consecutive memory addresses.

Reset. A signal that forces a CPU to enter its startup mode.

ROM. Read-Only Memory; a memory containing fixed data that is permanently defined as part of the manufacturing process. A CPU can use the information in the ROM, but cannot change it.

RS-232. A standard serial interface defined by the Electronic Industries Association (EIA).

Serial. One bit at a time.

Serial-access (or sequential-access). Refers to devices from which data can be retrieved only by passing through several locations between the one currently being accessed and the desired one. A tape is a typical serial-access device.

Shift Register. A clocked device that moves its contents one bit to the left or right during each clock cycle.

Single-user. Capable of handling only one user at a time.

Smart or intelligent terminal. A terminal that has computing capability of its own and is not totally dependent on the computer to which it is attached.

Software. Computer or microprocessor programs.

Source file. A file that contains statements written in a computer language. A source file must be assembled, compiled, or interpreted before it can be executed.

Stack. A set of tasks, storage addresses, or other items that are used in a last-in, first-out manner; that is, the last item entered is the first to be removed.

Static memory. A memory that does not lose its contents without external causes (as opposed to a dynamic memory).

String. An array (set of data) consisting of characters.

String functions. Procedures that allow the programmer to operate on data consisting of characters rather than numbers. Typical functions are insertion, deletion, concatenation, search, and replacement.

Structured programming. A programming method in which all programs consist of structures from a limited but complete set; each structure has a single entry and a single exit to simplify debugging.

Subroutine. A program that is subordinate to another program. The process whereby the computer transfers control to the subroutine is referred to as a subroutine call. Subroutine linkage is the mechanism used to transfer control to and from the subroutine.

Symbol table. A table defining all the names and labels used in a program.

Synchronous. Operating at regular time intervals according to a clock.

Syntax. Rules governing the structure of language. The language may be either human (natural) or a computer language.

Systems program. A program that is part of the operating system.

Systems software. Programs that perform administrative functions or aid in the development of other programs but do not actually perform any of the computer's ultimate workload.

Task. A program that the computer can execute.

Terminal. An input/output device used to enter data into a computer and record the output; usually consists of a keyboard and a printer or video display.

Time-sharing. Providing service to many users by working on each person's task part of the time.

Top-down design. A method for designing computer programs in which the programmer designs, codes, and tests the overall structure first and subsequently defines parts of the structure in increasingly greater detail.

Trace. A debugging aid that provides information about a program while it is executing. The trace prints all or some of the intermediate results.

TTL. Transistor-transistor logic; a popular type of integrated circuit.

Universal Asynchronous Receiver/Transmitter (UART). An LSI device that acts as an interface between systems. It handles data in parallel as well as devices that handle data in asynchronous serial form.

Universal Synchronous Receiver/Transmitter (USRT). Like a UART but handles data in synchronous serial form.

User program. A program that performs a task specified by the ultimate user of the computer.

User-programmable. The user can determine the purpose of the feature by writing an appropriate program.

Utility program. A program that performs a basic systems task, such as loading and saving programs, initiating program execution, observing and changing the contents of memory locations, or converting programs from one form to another.

Vectored interrupt. An interrupt that directs the computer to the specific service routine required to respond to it.

Video RAM. Separate memory that is used to hold information being shown on a video display.

Volatile memory. A memory that loses its contents when power is removed. Most RAM is volatile.

Winchester disk. A type of sealed hard-disk memory.

Word. The characteristic bit length of a computer, usually the length of its data paths, registers, and arithmetic unit.

Workspace. An area of memory or backup storage assigned for temporary use. Also called a workfile.

Write protect. A feature that prevents the computer from writing into an area of memory or storage. This protects the area from accidental overwriting or unauthorized changes.

APPENDIX B

INTEGRATED-CIRCUIT TECHNOLOGIES

Microprocessors and support integrated-circuit devices are made using a variety of fabrication methods or technologies. Each of these technologies has relative advantages and disadvantages. The two main semiconductor technologies are Metal-Oxide Semiconductor (MOS) and Bipolar. These terms refer to the type of transistor used as the active element in the device. MOS devices are typically used for microprocessors, microprocessor I/O devices, semiconductor memories, and other large-scale integration (LSI) devices. Bipolar technology is used in special-purpose microprocessors and in microprocessor peripheral devices that have a primary emphasis on functional speed. Bipolar technology is also used for the fabrication of simple Small Scale Integration (SSI) chips used to connect microprocessor-based system devices.

There are several variations within these two main technology groups. This Appendix provides examples of the more popular technologies used in the design of a microprocessor-based system. Some indications of the performance of these technologies are comparisons of average gate delay, power dissipation, and speed-power product. The speed-power product is a general figure of merit of the functional performance. Table B-1 and Figures B-1 and B-2 represent the general performance of the more popular integrated-circuit technologies. The power dissipation of CMOS devices is directly proportional to the clock frequency; bipolar devices are not as sensitive to clock frequency.

Table B-1. Selected integrated-circuit technologies and performance levels.

Technology		Speed-Power Product (Picojoules)	Average Gate: Propagation Delay (Nanoseconds)	Average Gate: Power Dissipation (Milliwatts)
BIPOLAR				
Transistor-Transistor Logic (TTL)				
1. Standard	(STD-TTL)	100	10	10
2. Low Power	(LP-TTL)	33	33	1
3. Schottky	(S-TTL)	57	3	19
4. Low-Power Schottky	(LS-TTL)	22	11	2
5. Advance Schottky	(AS-TTL)	13.6	1.7	8
6. Advance Low-Power Schottky	(ALS-TTL)	4.8	4	1.2
Emitter-Coupled Logic (ECL)				
1. MECL III		60	1	60
2. MECL 10K		50	2	25
3. MECL 10K H		25	1	25
Integrated Injection Logic (I^2L)		0.75	15	0.05
MOS				
N-channel MOS (NMOS)		5	25	0.2
Complementary MOS (CMOS)		0.04	40	0.0001*
High Speed CMOS (HCMOS)		0.008	8	0.0001* 0.5**

*Static Clock Condition
**With a 100-kHz Clock

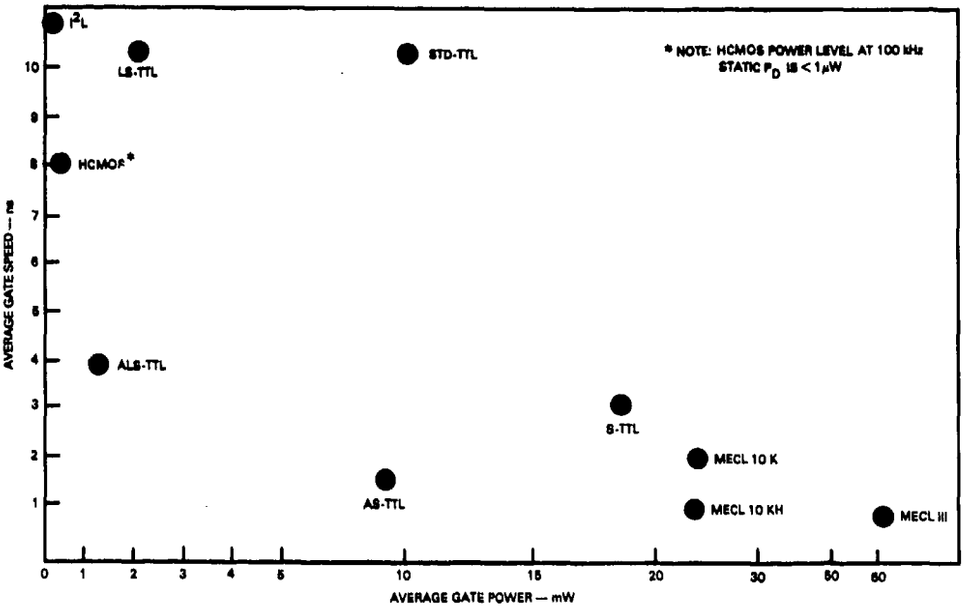


Figure B-1. Integrated-circuit technologies and performance.

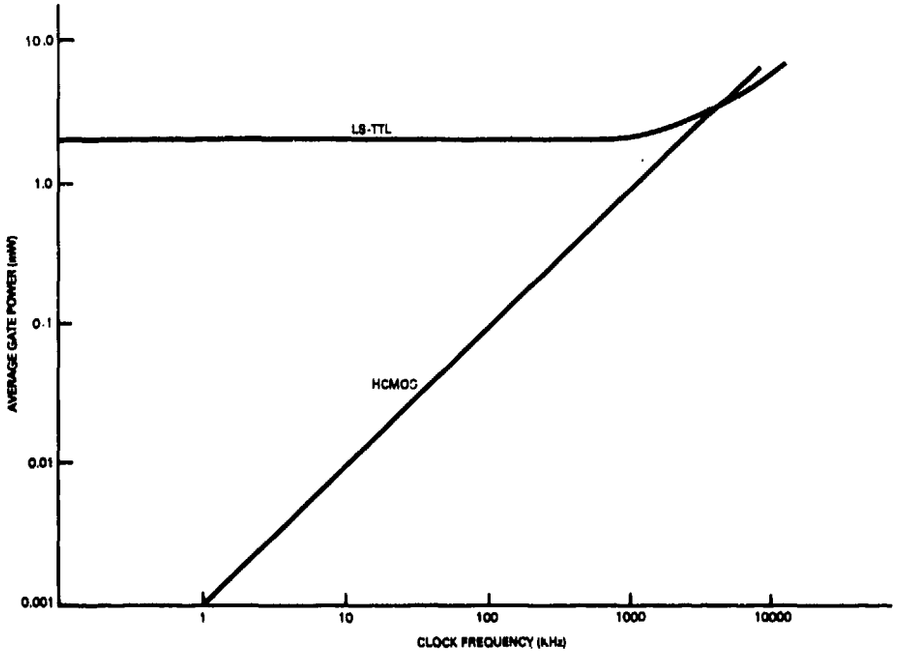


Figure B-2. Average gate power as a function of clock frequency.

Annex 1

AGARD FLIGHT TEST INSTRUMENTATION AND FLIGHT TEST TECHNIQUES SERIES

1. Volumes in the AGARD Flight Test Instrumentation Series, AGARDograph 160

Volume Number	*Title*	*Publication Date*
1.	Basic Principles of Flight Test Instrumentation Engineering by A.Pool and D.Bosman (to be revised in 1989)	1974
2.	In-Flight Temperature Measurements by F.Trenkle and M.Reinhardt	1973
3.	The Measurement of Fuel Flow by J.T.France	1972
4.	The Measurement of Engine Rotation Speed by M.Vedrunes	1973
5.	Magnetic Recording of Flight Test Data by G.E.Bennett	1974
6.	Open and Closed Loop Accelerometers by I.Mclaren	1974
7.	Strain Gauge Measurements on Aircraft by E.Kottkamp, H.Wilhelm and D.Kohl	1976
8.	Linear and Angular Position Measurement of Aircraft Components by J.C.van der Linden and H.A.Mensink	1977
9.	Aeroelastic Flight Test Techniques and Instrumentation by J.W.G.van Nunen and G.Piazzoli	1979
10.	Helicopter Flight Test Instrumentation by K.R.Ferrell	1980
11.	Pressure and Flow Measurement by W.Wuest	1980
12.	Aircraft Flight Test Data Processing — A Review of the State of the Art by L.J.Smith and N.O.Matthews	1980
13.	Practical Aspects of Instrumentation System Installation by R.W.Borek	1981
14.	The Analysis of Random Data by D.A.Williams	1981
15.	Gyroscopic Instruments and their Application to Flight Testing by B.Stieler and H.Winter	1982
16.	Trajectory Measurements for Take-off and Landing Test and Other Short-Range Applications by P.de Benque d'Agut, H.Riebeek and A.Pool	1985
17.	Analogue Signal Conditioning for Flight Test Instrumentation by D.W.Veatch and R.K.Bogue	1986

Volume Number	*Title*	*Publication Date*
18.	Microprocessor Applications inairborne flight Test Instrumentation by M.J.Prickett	1987

At the time of publication of the present volume the following volume was in preparation:

Digital Signal Conditioning for Flight Test Instrumentation
by G.A.Bever

2. Volumes in the AGARD Flight Test Techniques Series

Number	*Title*	*Publication Date*
AG 237	Guide to In-Flight Thrust Measurement of Turbojets and Fan Engines by the MIDAP Study Group (UK)	1979

The remaining volumes will be published as a sequence of Volume Numbers of AGARDograph 300.

Volume Number	*Title*	*Publication Date*
1.	Calibration of Air-Data Systems and Flow Direction Sensors by J.A.Lawford and K.R.Nippress	1983
2.	Identification of Dynamic Systems by R.E.Maine and K.W.Iliff	1985
3.	Identification of Dynamic Systems – Applications to Aircraft Part 1: The Output Error Approach by R.E.Maine and K.W.Iliff	1986
4.	Determination of Antenna Patterns and Radar Reflection Characteristics of Aircraft by H.Bothe and D.Macdonald	1986
5.	Store Separation Flight Testing by R.J.Arnold and C.S.Epstein	1986

At the time of publication of the present volume the following volumes were in preparation:

Identification of Dynamic Systems. Applications to Aircraft
Part 2: Nonlinear Model Analysis and Manoeuvre Design
by J.A.Mulder and J.H.Breeman

Flight Testing of Digital Navigation and Flight Control Systems
by F.J.Abbink and H.A.Timmers

Technqiues and Devices Applied in Developmental Airdrop Testing
by H.J.Hunter

Aircraft Noise Measurement and Analysis Techniques
by H.H.Heller

Air-to-Air Radar Flight Testing
by R.E.Scott

The Use of On-Board Computers in Flight Testing
by R.Langlade

Flight Testing under Extreme Environmental Conditions
by C.L.Hendrickson

Flight Testing of Terrain Following Systems
by C.Dallimore and M.K.Foster

Annex 2

AVAILABLE FLIGHT TEST HANDBOOKS

This annex is presented to make readers aware of handbooks that are available on a variety of flight test subjects not necessarily related to the contents of this volume.

Requests for A & AEE documents should be addressed to the Defence Research Information Centre, Glasgow (see back cover). Requests for US documents should be addressed to the Defense Technical Information Center, Cameron Station, Alexandria, VA 22314 (or in one case, the Library of Congress).

Number	*Author*	*Title*	*Date*
NATC-TM76-1SA	Simpson, W.R.	Development of a Time-Variant Figure-of-Merit for Use in Analysis of Air Combat Maneuvering Engagements	1976
NATC-TM76-3SA	Simpson, W.R.	The Development of Primary Equations for the Use of On-Board Accelerometers in Determining Aircraft Performance	1977
NATC-TM-77-1RW	Woomer, C. Carico, D.	A Program for Increased Flight Fidelity in Helicopter Simulation	1977
NATC-TM-77-2SA	Simpson, W.R. Oberle, R.A.	The Numerical Analysis of Air Combat Engagements Dominated by Maneuvering Performance	1977
NATC-TM-77-1SY	Gregoire, H.G.	Analysis of Flight Clothing Effects on Aircrew Station Geometry	1977
NATC-TM-78-2RW	Woomer, G.W. Williams, R.L.	Environmental Requirements for Simulated Helicopter/ VTOL Operations from Small Ships and Carriers	1978
NATC-TM-78-1RW	Yeend, R. Carico, D.	A Program for Determining Flight Simulator Field-of-View Requirements	1978
NATC-TM-79-33SA	Chapin, P.W.	A Comprehensive Approach to In-Flight Thrust Determination	1980
NATC-TM-79-3SY	Schiflett, S.G. Loikith, G.J.	Voice Stress Analysis as a Measure of Operator Workload	1980
NWC-TM-3485	Rogers, R.M.	Six-Degree-of-Freedom Store Program	1978
WSAMC-AMCP 706-204	—	Engineering Design Handbook, Helicopter Performance Testing	1974
NASA-CR-3406	Bennett, R.L. and Pearsons, K.S.	Handbook on Aircraft Noise Metrics	1981
—	—	Pilot's Handbook for Critical and Exploratory Flight Testing. (Sponsored by AIAA & SETP — Library of Congress Card No. 76-189165)	1972
—	—	A & AEE Performance Division Handbook of Test Methods for Assessing the Flying Qualities and Performance of Military Aircraft. Vol.1 Airplanes	1979
A & AEE Note 2111	Appleford, J.K.	Performance Division: Clearance Philosophies for Fixed Wing Aircraft	1978

Number	*Author*	*Title*	*Date*
A & AEE Note 2113 (Issue 2)	Norris, E.J.	Test Methods and Flight Safety Procedures for Aircraft Trials Which May Lead to Departures from Controlled Flight	1980
AFFTC-TD-75-3	Mahlum, R.	Flight Measurements of Aircraft Antenna Patterns	1973
AFFTC-TIH-76-1	Reeser, K. Brinkley, C. and Plews, L.	Inertial Navigation Systems Testing Handbook	1976
AFFTC-TIH-79-1	—	USAF Test Pilot School (USAFTPS) Flight Test Handbook. Performance: Theory and Flight Techniques	1979
AFFTC-TIH-79-2	—	USAFTPS Flight Test Handbook. Flying Qualities: Theory (Vol.1) and Flight Test Techniques (Vol.2)	1979
AFFTC-TIM-81-1	Rawlings, K., III	A Method of Estimating Upwash Angle at Noseboom-Mounted Vanes	1981
AFFTC-TIH-81-1	Plews, L. and Mandt, G.	Aircraft Brake Systems Testing Handbook	1981
AFFTC-TIH-81-5	DeAnda, A.G.	AFFTC Standard Airspeed Calibration Procedures	1981
AFFTC-TIH-81-6	Lush, K.	Fuel Subsystems Flight Test Handbook	1981
AFEWC-DR 1-81	—	Radar Cross Section Handbook	1981
NATC-TM-71-ISA226	Hewett, M.D. Galloway, R.T.	On Improving the Flight Fidelity of Operational Flight/ Weapon System Trainers	1975
NATC-TM-TPS76-1	Bowes, W.C. Miller, R.V.	Inertially Derived Flying Qualities and Performance Parameters	1976
NASA Ref. Publ. 1008	Fisher, F.A. Plumer, J.A.	Lightning Protection of Aircraft	1977
NASA Ref. Publ. 1046	Gracey, W.	Measurement of Aircraft Speed and Altitude	1980
NASA Ref. Publ. 1075	Kalil, F.	Magnetic Tape Recording for the Eighties (Sponsored by: Tape Head Interface Committee)	1982

The following handbooks are written in French and are edited by the French Test Pilot School (EPNER Ecole du Personnel Navigant d'Essais et de Réception ISTRES — FRANCE), to which requests should be addressed.

Number EPNER Reference	*Author*	*Title*	*Price (1983) French Francs*	*Notes*
2	G.Leblanc	L'analyse dimensionnelle	20	Réédition 1977
7	EPNER	Manuel d'exploitation des enregistrements d'Essais en vol	60	6ème Edition 1970
8	M.Durand	La mécanique du vol de l'hélicoptère	155	1ère Edition 1981
12	C.Laburthe	Mécanique du vol de l'avion appliquée aux essais en vol	160	Réédition en cours
15	A.Hisler	La prise en main d'un avion nouveau	50	1ère Edition 1964
16	Candau	Programme d'essais pour l'évaluation d'un hélicoptère et d'un pilote automatique d'hélicoptère	20	2ème Edition 1970
22	Cattaneo	Cours de métrologie	45	Réédition 1982

Number EPNER Reference	*Author*	*Title*	*Price (1983) French Francs*	*Notes*
24	G.Fraysse F.Cousson	Pratique des essais en vol (en 3 Tomes)	T 1 = 160 T 2 = 160 T 3 = 120	1ère Edition 1973
25	EPNER	Pratique des essais en vol hélicoptère (en 2 Tomes)	T 1 = 150 T 2 = 150	Edition 1981
26	J.C.Wanner	Bang sonique	60	
31	Tarnowski	Inertie-verticale-sécurité	50	1ère Edition 1981
32	B.Pennacchioni	Aéroélasticité — le flottement des avions	40	1ère Edition 1980
33	C.Lelaie	Les vrilles et leurs essais	110	Edition 1981
37	S.Allenic	Electricité à bord des aéronefs	100	Edition 1978
53	J.C.Wanner	Le moteur d'avion (en 2 Tomes) T 1 Le réacteur ..... T 2 Le turbopropulseur	 85 85	Réédition 1982
55	De Cennival	Installation des turbomoteurs sur hélicoptères	60	2ème Edition 1980
63	Gremont	Aperçu sur les pneumatiques et leurs propriétés	25	3ème Edition 1972
77	Gremont	L'atterrissage et le problème du freinage	40	2ème Edition 1978
82	Auffret	Manuel de médicine aéronautique	55	Edition 1979
85	Monnier	Conditions de calcul des structures d'avions	25	1ère Edition 1964
88	Richard	Technologie hélicoptère	95	Réédition 1971

REPORT DOCUMENTATION PAGE			
1. Recipient's Reference	**2. Originator's Reference** AGARD-AG-160 Volume 18	**3. Further Reference** ISBN 92-835-1542-0	**4. Security Classification of Document** UNCLASSIFIED

5. Originator Advisory Group for Aerospace Research and Development
North Atlantic Treaty Organization
7 rue Ancelle, 92200 Neuilly sur Seine, France

6. Title
MICROPROCESSOR APPLICATIONS IN AIRBORNE FLIGHT TEST INSTRUMENTATION

7. Presented at

8. Author(s)/Editor(s)	**9. Date**
M.J.Prickett and Edited by R.K.Bogue	February 1987
10. Author's/Editor's Address	**11. Pages**
Various	64

12. Distribution Statement This document is distributed in accordance with AGARD policies and regulations, which are outlined on the Outside Back Covers of all AGARD publications.

13. Keywords/Descriptors

Flight tests	Airborne computers
Avionics	Aircraft engines

14. Abstract

This volume in the AGARD Flight Test Instrumentation Series addresses flight test engineers and flight test instrumentation engineers interested in the design of microprocessors into new airborne flight test equipment. It describes general microprocessor based, system design principles and architectures suitable for flight test and flight test instrumentation applications. In preparing this volume the author has met with several engineers who are actively participating in aircraft and electronic flight testing at various organisations. Each of these organisations has developed microprocessor based instrumentation to solve their unique requirements and these are described in this text as case studies of current microprocessor applications.

This AGARDograph has been sponsored by the Flight Mechanics Panel of AGARD.

AGARD AGARDograph No.160 Vol.18 Advisory Group for Aerospace Research and Development, NATO MICROPROCESSOR APPLICATIONS IN AIRBORNE FLIGHT TEST INSTRUMENTATION by M.J.Prickett, edited by R.K.Bogue Published February 1987 64 pages This volume in the AGARD Flight Test Instrumentation Series addresses flight test engineers and flight test instrumentation engineers interested in the design of microprocessors into new airborne flight test equipment. It describes general microprocessor based, system design principles and architectures suitable for flight test and flight test instrumentation applications. In preparing this P.T.O	AGARD-AG-160 Volume 18 Flight tests Avionics Airborne computers Aircraft engines	AGARD AGARDograph No.160 Vol.18 Advisory Group for Aerospace Research and Development, NATO MICROPROCESSOR APPLICATIONS IN AIRBORNE FLIGHT TEST INSTRUMENTATION by M.J.Prickett, edited by R.K.Bogue Published February 1987 64 pages This volume in the AGARD Flight Test Instrumentation Series addresses flight test engineers and flight test instrumentation engineers interested in the design of microprocessors into new airborne flight test equipment. It describes general microprocessor based, system design principles and architectures suitable for flight test and flight test instrumentation applications. In preparing this P.T.O	AGARD-AG-160 Volume 18 Flight tests Avionics Airborne computers Aircraft engines
AGARD AGARDograph No.160 Vol.18 Advisory Group for Aerospace Research and Development, NATO MICROPROCESSOR APPLICATIONS IN AIRBORNE FLIGHT TEST INSTRUMENTATION by M.J.Prickett, edited by R.K.Bogue Published February 1987 64 pages This volume in the AGARD Flight Test Instrumentation Series addresses flight test engineers and flight test instrumentation engineers interested in the design of microprocessors into new airborne flight test equipment. It describes general microprocessor based, system design principles and architectures suitable for flight test and flight test instrumentation applications. In preparing this P.T.O	AGARD-AG-160 Volume 18 Flight tests Avionics Airborne computers Aircraft engines	AGARD AGARDograph No.160 Vol.18 Advisory Group for Aerospace Research and Development, NATO MICROPROCESSOR APPLICATIONS IN AIRBORNE FLIGHT TEST INSTRUMENTATION by M.J.Prickett, edited by R.K.Bogue Published February 1987 64 pages This volume in the AGARD Flight Test Instrumentation Series addresses flight test engineers and flight test instrumentation engineers interested in the design of microprocessors into new airborne flight test equipment. It describes general microprocessor based, system design principles and architectures suitable for flight test and flight test instrumentation applications. In preparing this P.T.O	AGARD-AG-160 Volume 18 Flight tests Avionics Airborne computers Aircraft engines

volume the author has met with several engineers who are actively participating in aircraft and electronic flight testing at various organisations. Each of these organisations has developed microprocessor based instrumentation to solve their unique requirements and these are described in this text as case studies of current microprocessor applications.

This AGARDograph has been sponsored by the Flight Mechanics Panel of AGARD.

ISBN 92-835-1542-0

volume the author has met with several engineers who are actively participating in aircraft and electronic flight testing at various organisations. Each of these organisations has developed microprocessor based instrumentation to solve their unique requirements and these are described in this text as case studies of current microprocessor applications.

This AGARDograph has been sponsored by the Flight Mechanics Panel of AGARD.

ISBN 92-835-1542-0

volume the author has met with several engineers who are actively participating in aircraft and electronic flight testing at various organisations. Each of these organisations has developed microprocessor based instrumentation to solve their unique requirements and these are described in this text as case studies of current microprocessor applications.

This AGARDograph has been sponsored by the Flight Mechanics Panel of AGARD.

ISBN 92-835-1542-0

volume the author has met with several engineers who are actively participating in aircraft and electronic flight testing at various organisations. Each of these organisations has developed microprocessor based instrumentation to solve their unique requirements and these are described in this text as case studies of current microprocessor applications.

This AGARDograph has been sponsored by the Flight Mechanics Panel of AGARD.

ISBN 92-835-1542-0